多人通信框架

与该框架下的扩展规则五子棋游戏

代码手册

周子凯 1400012702

黎才华 1400012776

唐毅 1400012711

目录

类 framework.InvitationCode

继承自： Object

InvitationCode 类提供三个静态方法，实现网络地址（IP 地址、端口号）与邀请码之间的转换。

方法	
static String	getCode(InetAddress addr, int port) 返回 addr 和 port 所对应的邀请码 **参数：** addr - 主机的 IP 地址 port - 端口号 **返回：** 给定 addr 和 port 所对应的邀请码
static InetAddress	getInetAddress(String code) 返回 code 所对应的 IP 地址 **参数：** code - 邀请码 **返回：** 给定邀请码所对应的 IP 地址
static int	getPort(String code) 返回 code 所对应的端口号 port **参数：** code - 邀请码 **返回：** 给定邀请码所对应的端口号

类 framework.SocketStream

继承自： Object

SocketStream 类用 ObjectInputStream 和 ObjectOutputStream 对给定 Socket 类实例的输入输出流进行包装，提供该 Socket 上的对象传输功能及通信时的互斥锁。

字段	
ObjectInputStream	istream 用于对所绑定的 Socket 的输入流进一步包装
ObjectOutputStream	ostream 用于对所绑定的 Socket 的输出流进一步包装

方法	
boolean	bind(Socket socket) 与某个 Socket 进行绑定，用 istream 和 ostream 分别包装该 socket 的输入输出流。 **参数：** socket – 给定套接字 **返回：** 绑定是否成功
void	acquire() 获取使用通信的许可，在获取许可前一直阻塞。
void	release() 释放使用通信的许可。
void	close() 关闭绑定的 socket 并解除绑定。
boolean	isValid() 查询该对象是否已绑定某个 Socket。 **返回：** 该对象是否已绑定某个 Socket。

类 framework.ObjectStream

继承自： Object

ObjectStream 类提供三个静态方法，在给定 SocketStream 类实例上完成对象通信。

方法	
static <T> T	recv(SocketStream s) 在 SocketStream 上接收一个类型为 T 的对象。 **参数：** s – 给定的 SocketStream 类实例 **返回：** 在 s 上接收的一个类型 T 的对象。
static void	send(SocketStream s, Object... objects) 在 SocketStream 上发送若干对象。 **参数：** s – 给定的 SocketStream 类实例 objects – 要发送的若干对象
static void	sendWithLock(SocketStream s, Object... objects) 在 SocketStream 上发送若干对象并加锁互斥锁。 **参数：** s – 给定的 SocketStream 类实例 objects – 要发送的若干对象

类 framework.Client

继承自：Object
子类：litz.Client

Client 类为平台层的客户端类，用于与服务端建立连接。

方法	
boolean	connect(String serverCode) 通过邀请码与对应的服务端建立连接。 **参数：** serverCode – 邀请码 **返回：** 连接建立是否成功
void	close() 关闭与服务端的连接。

类 framework.Server

继承自： Object
子类： litz.Server

Server 类为平台层的服务端类，用于与多个客户端建立连接。

方法	
void	ready() 内部初始化。
string	getCode() 获取内部监听端口对应的邀请码。 **返回：** 内部监听端口对应的邀请码。
void	listen() 开始监听客户端的连接请求。
void	finish() 结束监听。
void	close(int clientIndex) 关闭与指定编号的客户端之间的连接。 **参数：** clientIndex – 指定客户端的编号
void	close() 关闭监听且关闭与所有客户端之间的连接。
protected int	maxClientNumber() 获取允许的最大客户端连接数，负数表示没有限制。本类中返回-1，可由派生类重写。 **返回：** 允许的最大客户端连接数，负数表示没有限制
protected void	onConnect(SocketStream client) 建立与客户端之间的连接时的回调。本类中直接返回，可由派生类重写。 **参数：** client – 建立连接的 SocketStream 类实例
protected void	onArrayed(int clientIndex) 将建立连接的客户端加入客户端数组时的回调。本类中直接返回，可由派生类重写。 **参数：** clientIndex – 建立连接的客户端在客户端数组中的下标

类 litz.Point

继承自： Object
实现： Serializable

Point 类代表棋盘上的点。Point 类为不可变类。

字段	
final byte	x 点在棋盘上的横坐标。
final byte	y 点在棋盘上的纵坐标。

构造函数

Point(byte x, byte y)
用横纵坐标构造对象。
参数：
x – 横坐标
y – 纵坐标

方法	
Point	move(Vector displacement) 创建一个新的 Point 类实例，其坐标为当前坐标加上给定位移。 **参数：** displacement – 给定位移 **返回：** 新的 Point 类实例，其坐标为当前坐标加上给定位移
Point	copy() 创建一个新的 Point 类实例，其坐标与当前坐标相同。 **返回：** 新的 Point 类实例，其坐标与当前坐标相同

类 litz.Vector

继承自： Object
实现： Serializable

Vector 类代表棋盘上的点之间的位移。Vector 类为不可变类。

字段	
final byte	dx 点之间的横坐标的位移。
final byte	dy 点之间的纵坐标的位移。

构造函数
Vector(byte dx, byte dy) 用横纵坐标位移构造对象。 **参数：** x – 横坐标位移 y – 纵坐标位移

方法	
Vector	reverse() 创建一个新的 Vector 类实例，其位移为当前位移的负向。 **返回：** 新的 Vector 类实例，其位移为当前位移的负向
Vector	copy() 创建一个新的 Vector 类实例，其位移与当前位移相同。 **返回：** 新的 Vector 类实例，其位移与当前位移相同

类 litz.Piece

继承自： Object
实现： Serializable

Piece 类代表棋盘上的棋子。Piece 类为不可变类。

字段	
final Point	position 棋子在棋盘上的位置。
final byte	styleIndex 棋子的样式编号。

构造函数

Piece(Point position, byte pieceStyleIndex)
用位置与样式编号构造对象。
参数：
position – 位置
pieceStyleIndex – 样式编号

类 litz.Player

继承自： Object
实现： Serializable

Player 类代表游戏玩家信息。

字段	
static final int	MAX_STYLE_NUMBER 最大棋子样式总数。
static final int	MAX_PLAYER_NUMBER 最大玩家总数。
byte	clientIndex 玩家所在的客户端编号。
volatile boolean	isHuman 玩家是否为人类玩家（机器玩家）。
byte	pieceStyleIndex 玩家使用的棋子样式编号。
String	name 玩家名。

构造函数

Player(byte clientIdx, byte pieceStyleIdx, String playerName)
用客户端编号、棋子样式和玩家名构造人类玩家对象。

参数：

clientIdx – 客户端编号
pieceStyleIdx – 棋子样式
playerName – 玩家名

类 litz.Rule

继承自： Object
实现： Serializable

Rule 类代表游戏规则配置信息。

字段	
byte	connectNumber 游戏的连子数。
boolean	winByConnect 游戏是否通过连子判断胜利。
boolean	allowOverline 游戏是否支持长连。
boolean	allowSlant 游戏是否支持斜连。

构造函数

Rule()
用缺省配置构造对象。等价于 Rule(5, true, true, true)

Rule(byte connectNumber, boolean winByConnect, boolean allowOverline, boolean allowSlant)
用所有四个配置构造对象。

参数：

connectNumber – 连子数
winByConnect – 是否通过连子判断胜利
allowOverline – 是否支持长连
allowSlant – 是否支持斜连

类 litz.Setting

继承自： Object
实现： Serializable

Setting 类代表游戏配置信息。

字段	
Rule	rule 游戏规则配置信息。
Player[]	players 游戏玩家信息。
byte	boardSize 游戏的棋盘边长。

构造函数
Setting() 用缺省配置构造对象。游戏规则配置为缺省配置，游戏玩家信息为空，棋盘边长为连子数的 4 倍。

类 litz.Board

继承自： Object

Board 类代表棋盘，存储棋子分布信息，提供对棋子的操作。

字段	
byte	size 棋盘大小。

构造函数
Board(byte boardSize) 用棋盘大小构造对象。 **参数：** boardSize – 棋盘大小

方法	
final void	clear() 清空棋盘。
void	clearPosition(Point position) 清空棋盘的指定位置。 **参数：** position – 指定位置
byte	get(Point position) 获取棋盘指定位置的棋子的样式编号。 **参数：** position – 指定位置 **返回：** 指定位置的棋子的样式编号，负数表示没有棋子
void	place(Piece piece) 在棋盘上放置指定棋子。 **参数：** piece – 指定棋子
void	remove(Piece piece) 从棋盘上取走指定棋子。 **参数：** piece – 指定棋子
boolean	isValid(Point position) 检查指定位置是否为棋盘上的合法位置。 **参数：** position – 指定位置 **返回：** 指定位置是否为棋盘上的合法位置

boolean	isBlank(Point position) 检查棋盘指定位置是否为空。 **参数：** position - 指定位置 **返回：** 棋盘指定位置是否为空
boolean	hasPiece(Piece piece) 检查棋盘上是否有指定棋子。 **参数：** piece - 指定棋子 **返回：** 棋盘上是否有指定棋子

类 litz.Connection

继承自： Object

Connection 类代表棋盘上某个棋子在某个方向上的连子情况信息，通常以数组的方式使用，存储各个棋子在各个方向上的连子情况信息。

字段	
final ArrayList<Point>	connected 直接相连的各个同样式的棋子的位置。
final ArrayList<Point>	separated 不直接相连，但中间只间隔空位的各个棋子的位置。
final ArrayList<Point>	blank 间隔的空位。
final ArrayList<Point>	blankOut 棋子之外的空位。

构造函数
Connection() 构造对象。各个数组均为空。

类 litz.Game

继承自： Object

Game 类代表一次游戏，存储游戏配置、玩家信息、棋盘信息、游戏历史信息等，提供有关游戏的各种操作。

字段	
Rule	rule 游戏规则配置信息。
Player[]	players 游戏玩家信息。
Board	board 游戏的棋盘。

构造函数
Game(Setting setting) 用给定游戏配置构造对象。rule 字段与 players 字段与给定游戏配置中的相应字段相同，board 字段用给定游戏配置中的 boardSize 字段构造。 **参数：** setting – 给定游戏配置

方法	
byte	getPlayerNumber() 获取游戏玩家人数。 **返回：** 游戏玩家人数
boolean	canPlace(Piece piece) 检查是否可以在棋盘上下指定棋子。只能在空位下棋，若配置不支持长连，则若下在空位后达成长连，则不允许下在该空位。 **参数：** piece – 指定棋子 **返回：** 是否可以在棋盘上下指定棋子
void	place(Piece piece) 在棋盘上下指定棋子。 **参数：** piece – 指定棋子
void	undo() 取消上一次落子。

boolean	win(Point position) 检查棋盘指定位置的棋子是否达成胜利条件。若达成配置连子数或配置支持长连且达成大于等于配置连子数则胜利；若配置不通过连子判断胜利，则若达成关键位置胜利条件则胜利。 **参数：** position – 指定位置 **返回：** 棋盘指定位置的棋子是否达成胜利条件
Connection[][]	getConnections(Point position) 获取指定位置的连子情况信息。数组的第一维是在指定位置假设的棋子样式编号，第二维是各个连子方向。 **参数：** position – 指定位置 **返回：** 指定位置的连子情况信息

类 litz.Client

继承自： framework.Client

Client 类代表游戏的客户端，提供游戏操作和与服务端通信等功能。通信由发送服务码（见类 litz.Server）与后续通信构成。

字段	
Setting	setting 游戏开始前的配置。
Game	game 游戏开始后的游戏实例。

构造函数
Client() 构造对象。各字段为空。

方法	
boolean	connect(String serverCode) 重写基类方法。通过邀请码与对应的服务端建立连接，连接后从服务端获取游戏配置与客户端编号。 **参数：** serverCode – 邀请码 **返回：** 连接建立是否成功
void	close() 重写基类方法。关闭与服务端的连接，关闭内部通信线程。
void	bind(UserInterface UI) 绑定给定用户接口。 **参数：** UI – 给定用户接口
void	editPlayer(String playerName, byte pieceStyleIndex) 告知服务端修改用户信息。 **参数：** playerName – 修改后的用户名 pieceStyleIndex – 修改后的用户棋子样式编号

void	editRule(byte connectNumber, boolean winByConnect, boolean allowOverline, boolean allowSlant) 告知服务端修改游戏规则配置。 **参数：** connectNumber – 修改后的连子数 winByConnect – 修改后的是否通过连子判断胜利 allowOverline – 修改后的是否支持长连 allowSlant – 修改后的是否支持斜连
void	editBoardSize(byte boardSize) 告知服务端修改棋盘边长。 **参数：** boardSize – 修改后的棋盘边长
void	startGame() 告知服务端开始游戏。
void	placePiece(byte x, byte y) 告知服务端在指定位置落子。 **参数：** x – 指定横坐标 y – 指定纵坐标
void	exitGame() 告知服务端退出游戏。
Connection[][]	getConnections(Point position) 获取指定位置的连子情况信息。同 Game 类。 **参数：** position – 指定位置 **返回：** 指定位置的连子情况信息

类 litz.Server

继承自： framework.Server

Server 类代表游戏的服务端，提供游戏逻辑处理和与客户端通信等功能。通信由发送服务码与后续通信构成。

字段	
static final int	SERVICE_EXIT 告知服务端退出游戏服务码。无后续通信。
static final int	SERVICE_START 告知服务端开始游戏服务码。无后续通信。
static final int	SERVICE_TURN 告知客户端开始回合服务码。无后续通信。
static final int	SERVICE_END 告知客户端结束游戏服务码。无后续通信。
static final int	SERVICE_PLACE 告知服务端落子服务码。后续通信：客户端发送 Position 类实例，表示落子位置。 告知客户端落子服务码。后续通信：服务端发送 Piece 类实例，表示落子。
static final int	SERVICE_INVALID 告知客户端落子无效服务码。无后续通信。
static final int	SERVICE_WIN 告知客户端某玩家胜利服务码。后续通信：服务端发送 Byte 类实例，表示胜利玩家编号。
static final int	SERVICE_EDIT_EXIT 告知客户端某玩家退出游戏服务码。后续通信：服务端发送 Byte 类实例，表示退出游戏玩家编号。
static final int	SERVICE_EDIT_ENTR 告知客户端某玩家加入游戏服务码。后续通信：服务端发送 Player 类实例，表示加入游戏玩家信息。
static final int	SERVICE_EDIT_PLYR 告知服务端修改玩家信息服务码。后续通信：客户端发送 String 类实例，表示所要修改的玩家名，再发送 Byte 类实例，表示所要修改的玩家棋子样式编号。 告知客户端修改玩家信息服务码。后续通信：服务端发送 Player 类实例，表示修改后的玩家信息。

static final int	SERVICE_EDIT_RULE 　　告知服务端修改游戏规则配置服务码。后续通信：客户端发送 Rule 类实例，表示所要修改的游戏规则配置。 　　告知客户端修改游戏规则配置服务码。后续通信：服务端发送 Rule 类实例，表示修改后的游戏规则配置。
static final int	SERVICE_EDIT_SIZE 　　告知服务端修改棋盘边长服务码。后续通信：客户端发送 Byte 类实例，表示所要修改的棋盘边长。 　　告知客户端修改棋盘边长服务码。后续通信：服务端发送 Byte 类实例，表示修改后的棋盘边长。
static final int	SERVICE_INDEX 　　告知服务端要求获取客户端编号服务码。无后续通信。 　　告知客户端客户端编号服务码。后续通信：服务端发送 Byte 类实例，表示客户端编号。

构造函数
Server() 　　构造对象。

构造函数	
void	ready() 　　重写基类方法。内部初始化。
void	close(int clientIndex) 　　重写基类方法。关闭与指定编号的客户端之间的连接，用机器玩家代替其进行游戏。 **参数：** 　　clientIndex – 指定客户端的编号
void	close() 　　重写基类方法。关闭监听且关闭与所有客户端之间的连接，关闭内部通信线程。
protected int	maxClientNumber() 　　重写基类方法。获取允许的最大客户端连接数。 **返回：** 　　允许的最大客户端连接数 Player.MAX_PLAYER_NUMBER
protected void	onConnect(SocketStream client) 　　重写基类方法。建立与客户端之间的连接时的回调。向连接的客户端发送游戏配置信息。 **参数：** 　　client – 建立连接的 SocketStream 类实例

protected void	onArrayed(int clientIndex) 重写基类方法。将建立连接的客户端加入客户端数组时的回调。创建内部通信线程，发送自动分配的玩家信息。 **参数：** clientIndex – 建立连接的客户端在客户端数组中的下标

类 abstract litz.UserInterface

继承自： Object

UserInterface 抽象类代表用户接口，连接游戏逻辑与 GUI 或 CLUI，主要包含 UI 需要调用的游戏功能（非抽象部分）与游戏的回调（抽象部分）。

字段	
protected Server	server UI 绑定的服务端，派生类可调用其功能。
protected Client	client UI 绑定的客户端，派生类可调用其功能。

方法	
abstract void	getIndex(byte index) 修改本客户端编号的回调。 **参数：** index – 修改的本客户端编号
abstract void	changePlayer(byte playerIndex, String playerName, byte pieceStyleIndex) 修改玩家信息的回调。 **参数：** playerIndex – 修改的玩家的编号 playerName – 修改的玩家名 pieceStyleIndex – 修改的棋子样式编号
abstract void	removePlayer(byte playerIndex) 删除玩家信息的回调。 **参数：** playerIndex – 删除的玩家的编号
abstract void	changeRule(byte connectNumber, boolean winByConnect, boolean allowOverline, boolean allowSlant) 修改游戏规则配置信息的回调。 **参数：** connectNumber – 修改的连子数 winByConnect – 修改的是否通过连子判断胜利 allowOverline – 修改的是否支持长连 allowSlant – 修改的是否支持斜连
abstract void	changeBoardSize(byte boardSize) 修改棋盘边长的回调。 **参数：** boardSize – 修改的棋盘边长

abstract void	next(byte playerIndex) 切换当前回合玩家的回调。 **参数：** playerIndex - 当前回合玩家的编号
abstract void	turn(boolean ownTurn) 切换本用户回合指示的回调。 **参数：** ownTurn - 是否为本用户回合
abstract void	place(byte x, byte y, byte pieceStyleIndex) 落子的回调。 **参数：** x - 落子的横坐标 y - 落子的纵坐标 pieceStyleIndex - 落子的棋子样式编号
abstract void	win(byte playerIndex) 某玩家胜利的回调。 **参数：** playerIndex - 胜利的玩家的编号
void	doQuit(boolean isForced) 退出游戏界面的回调。 **参数：** isForced - 本次退出是否不是正常游戏结束导致的强制退出
abstract void	forceQuit() 强制退出游戏界面的回调。
abstract void	back() 正常退出游戏界面的回调。

类 litz.TextDiv

继承自： JLabel

此类实现了用于显示文字的组件。组件采用绝对定位，并使用游戏的自定义字体。

构造方法
TextDiv(String text, int posx, int posy, int width, int height) 新建一个 TextDiv 的实例，显示指定的文字，具有指定的坐标和长宽。背景颜色为透明。 **参数：** text – 组件中显示的文字 posx – 组件左上角的 x 坐标 posy – 组件左上角的 y 坐标 width – 组件的宽，单位为像素 height – 组件的高，单位为像素
TextDiv(String text, int posx, int posy, int width, int height, Color col) 新建一个 TextDiv 的实例，显示指定的文字，具有指定的坐标和长宽以及背景颜色。 **参数：** text – 组件中显示的文字 posx – 组件左上角的 x 坐标 posy – 组件左上角的 y 坐标 width – 组件的宽，单位为像素 height – 组件的高，单位为像素 col – 组件的背景颜色

类 litz.Div

继承自： javax.swing.JPanel

子 类 ： litz.AnimatedBackground, litz.Button, litz.ButtonList, litz.ChessBoard, litz.GameView, litz.MainMenu, litz.NameCard, litz.Option, litz.OptionPanel, litz.PawnPanel

此类定义了组成程序界面的基本单元。一个 Div 类的实例是一个通过左上角坐标绝对定位的、具有指定长宽和纯色背景的矩形组件。

构造方法
Div() 创建一个 Div 的实例，未指定坐标和长宽。
Div(int posx, int posy, int width, int height) 创建一个 Div 的实例，具有指定坐标和长宽。背景色为透明。 **参数：** posx – 组件左上角的 x 坐标 posy – 组件左上角的 y 坐标 width – 组件的宽，单位为像素 height – 组件的高，单位为像素
Div(int posx, int posy, int width, int height, Color col) 创建一个 Div 的实例，具有指定坐标、长宽和背景色。 **参数：** posx – 组件左上角的 x 坐标 posy – 组件左上角的 y 坐标 width – 组件的宽，单位为像素 height – 组件的高，单位为像素 col – 组件的背景色

类 litz.NameCard

继承自： litz.Div

此类实现了游戏界面中玩家列表中每个玩家的信息栏，包含玩家的昵称和玩家控制的棋子颜色。玩家信息栏是游戏界面中表示玩家列表的 litz.ButtonList 实例的 list[]字段下的成员。

字段	
static int	height 玩家信息栏的高度，单位为像素。缺省值为游戏窗口高度的 7%。
static int	width 玩家信息栏的宽度，单位为像素。缺省值为游戏窗口宽度的 18%。
static int	margin 玩家信息栏周围留空边框的宽度，单位为像素。缺省值为游戏窗口高度的 1%。

构造方法

NameCard(int posx, int posy, int _playerID)

创建一个 NameCard 的实例，具有指定的坐标，显示指定编号玩家的信息。

参数：

posx – 组件左上角的 x 坐标

posy – 组件左上角的 y 坐标

_playerID – 显示信息的玩家编号（从 1 开始编号）

方法	
void	paintComponent(Graphics g) 定义本类实例的绘制方法，绘制玩家控制的棋子颜色以及玩家的昵称。不应该被主动调用。 **参数：** g – 此组件的图形上下文

类 litz.ButtonList

继承自： litz.Div
子类： litz.RoomPlayerList

此类实现了纵向堆叠的按钮组。当鼠标在按钮间移动时，鼠标所在的按钮的高亮背景框会跟随鼠标渐慢移动，同时鼠标所在的按钮会向右突出。按钮组中的每个按钮可以是 Div 类的任何子类，而不必要具有按钮的功能。在初始化 ButtonList 的一个实例后需要手动向该实例的 list[]字段中添加每个按钮。通过禁用 ButtonList 实例的 operatable 属性可将该实例转化为一个不可被用户操作的列表。

字段	
int	height 按钮组的高度，单位为像素。
int	width 按钮组的宽度，单位为像素。
int	buttonHeight 按钮组中每个按钮的高度，单位为像素。
int	buttonWidth 按钮组中每个按钮的高度，单位为像素。
Div[]	list 保存按钮组中每个按钮的数组。
static Image	selectorImage 鼠标所在按钮的高亮背景图片。
boolean	operatable 指示此按钮组是否会对鼠标的移入、点击做出反应。缺省为 true。

构造方法

ButtonList(int posx, int posy, int _nButton, int _buttonWidth, int _buttonHeight)
创建一个 ButtonList，具有指定的坐标和指定数量的按钮，每个按钮具有指定的长宽。
参数：
posx – 按钮组左上角的 x 坐标
posy – 按钮组左上角的 y 坐标
_nButton – 按钮组中按钮的数量
_buttonWidth – 每个按钮的宽，单位为像素
_buttonHeight – 每个按钮的高，单位为像素

方法	
void	moveSelector(int index) 将高亮背景框移动到第 index 个按钮后（从 0 开始标号）。 **参数：** index – 目标按钮的索引

类 litz.RoomPlayerList

继承自： litz.ButtonList

此类实现了游戏房间内的玩家列表。本类实例的 lits[] 字段中的每个元素是 litz.TextDiv 类的实例，或者 TextField 类的实例。列表内的每个条目代表一个玩家的昵称，其中用户本人的昵称在 TextField 组件中显示，可以通过修改其内容来修改自己的昵称；其它玩家的昵称在 litz.TextDiv 组件中显示，若其它玩家修改了自己了昵称，对应的 TextDiv 组件的内容将会更新。

字段	
int	myIndex 玩家的编号，编号从 0 开始。

构造方法

RoomPlayerList(int posx, int posy, int _buttonWidth, int _buttonHeight)

创建一个 RoomPlayerList 的实例，具有指定的坐标、长宽和棋盘行列坐标。

参数：

posx – 按钮组左上角的 x 坐标

posy – 按钮组左上角的 y 坐标

_buttonWidth – 每个按钮的宽，单位为像素

_buttonHeight – 每个按钮的高，单位为像素

方法	
void	initialize(int _myIndex) 初始化玩家列表。 **参数：** _myIndex – 用户的玩家编号，编号从 0 开始
void	updatePlayer(int index, String name) 为指定编号的玩家更新昵称 **参数：** index – 更新昵称的玩家的编号，编号从 0 开始 name – 更新后的昵称
void	reset() 将玩家列表重置为初始状态，为下轮游戏做准备。

类 litz.PawnPanel

继承自： litz.Div

此类实现了棋盘上的格点。一个 litz.ChessBoard 组件的实例下添加有多个 PawnPanel 组件的实例。用户点击未被占用的格点将向游戏内核发起落子请求。鼠标移动到一个格点内，棋盘将显示从此格点出发的、与邻近的棋子相连的辅助瞄准线和瞄准线经过的格点。

字段	
static Color[]	pawnColor 保存棋子可选的颜色的数组。最多支持 5 人同时游戏，此静态字段保存了 5 个 Color 类实例组成的数组，它们的值分别为 Color(150,230,209)， Color(86,152,241)， Color(179,119,251)， Color(243,103,111)， Color(245,217,55)。

构造方法

PawnPanel(int posx, int posy, int height, int width, int _co_x, int _co_y)

创建一个 PawnPanel 的实例，具有指定的坐标、长宽和棋盘行列坐标。

参数：

posx – 组件左上角的 x 坐标
posy – 组件左上角的 y 坐标
height – 组件的高，单位为像素
width – 组件的宽，单位为像素
_co_x – 此格点在棋盘上的行坐标（从 0 开始标号）
_co_y – 此格点在棋盘上的列坐标（从 0 开始标号）

方法	
void	paintComponent(Graphics g) 定义本类实例的绘制方法。如果该格点已被棋子占据，则绘制棋子。如果该棋子是某个玩家最后一次落的子，则绘制提示用边框。不应该被主动调用。 **参数：** g – 此组件的图形上下文

类 litz.Dot

继承自： Object

此类描述了棋盘上辅助瞄准线经过的格点，包含的信息有格点的行列坐标和颜色，其中颜色由该瞄准线上的其它棋子的拥有者和玩家-颜色的对应表决定。

字段	
int	x 格点的行号（从 0 开始编号）。
int	y 格点的列号（从 0 开始编号）。
Color	col 格点的颜色。

构造方法
Dot() 创建一个 Dot 的实例，未指定坐标和颜色。
Dot(int _x, int _y, int colorIndex) 创建一个 Dot 的实例，具有指定行列坐标和颜色。颜色通过给定标号的方式从玩家-颜色对应表的表项中选取。 **参数：** _x - 格点的行号（从 0 开始标号） _y - 格点的列号（从 0 开始标号） colorIndex - 格点的颜色在玩家-颜色表中的标号（从 0 开始标号）

类 litz.Line

继承自： Object

此类描述了棋盘上辅助瞄准线，包含的信息有瞄准线的两个端点的行列坐标，以及瞄准线的颜色，其中颜色由该瞄准线上的其它棋子的拥有者和玩家-颜色的对应表决定。

字段	
int	x1 辅助瞄准线起始格点的行号（从 0 开始编号）。
int	y1 辅助瞄准线起始格点的列号（从 0 开始编号）。
int	x2 辅助瞄准线终止格点的行号（从 0 开始编号）。
int	y2 辅助瞄准线终止格点的列号（从 0 开始编号）。
Color	col 辅助瞄准线的颜色

构造方法

Line()

创建一个 Line 的实例，未指定起止点坐标和颜色。

Line(int _x1, int _y1, int _x2, int _y2, int colorIndex)

创建一个 Line 的实例，具有执行的起止点坐标和颜色。颜色通过给定标号的方式从玩家-颜色对应表的表项中选取。

参数：

_x1 -辅助瞄准线起始格点的行号（从 0 开始编号）

_y1 - 辅助瞄准线起始格点的列号（从 0 开始编号）

_x2 - 辅助瞄准线终止格点的行号（从 0 开始编号）

_y2 - 辅助瞄准线终止格点的列号（从 0 开始编号）

colorIndex -辅助瞄准线的颜色在玩家-颜色表中的标号（从 0 开始标号）

类 litz.ChessBoard

继承自： litz.Div

此类实现了游戏界面中的棋盘。棋盘上的格点由 litz.PawnPanel 类实现。点击棋盘上的空的格点时，向 GUI 类发出在此位置落当前玩家的棋子的请求。当鼠标悬停在格点上时，棋盘绘制出从该点出发的、与邻近的棋子相连的辅助瞄准线，标记辅助瞄准线经过的格点。

字段	
int[][]	map 记录棋盘上每个格点的状态（为空或有某个玩家的棋子）的二维数组。若为 0 则表示该点为空，否则表示拥有该点的棋子的玩家编号（从 1 开始编号）。
int[][]	lastPawn 记录棋盘上每个格点上是否是某个玩家最后落的棋子。若不为 0，则表示该点有此编号的玩家最后一次落的子（玩家从 1 开始编号）。

构造方法
ChessBoard(int n, int m, int _nPlayer, int posx, int posy, int width) 创建一个 Chessboard，具有指定的行数、列数、玩家数量、坐标和长宽。棋盘的行数和列数可以不等，但棋盘外形为正方形。 **参数：** n – 棋盘的行数 m – 棋盘的列数 _nPlayer – 玩家人数 posx – 棋盘左上角的 x 坐标 posy – 棋盘左上角的 y 坐标 width – 棋盘的长和宽，单位为像素

方法	
void	putPawn(int x, int y, int player) 在第 x 行第 y 列落下玩家 player 的棋子（行列从 0 开始标号，玩家从 1 开始标号）。 **参数：** x – 落子位置的行号（从 0 开始标号） y – 落子位置的列号（从 0 开始标号） player – 落子玩家的行号（从 1 开始标号）
void	paintComponent(Graphics g) 定义本类实例的绘制方法，包括棋盘格线和辅助瞄准线的绘制。不应该被主动调用。 **参数：** g – 此组件的图形上下文

类 litz.GameView

继承自： litz.Div

此类实现了游戏界面。游戏界面包括玩家列表、“你的回合”提示栏、棋盘、退出按钮四部分。

字段	
litz.Button	buttonQuit 游戏界面中的退出按钮。
litz.TextDiv	yourTurn 游戏界面中的“你的回合”提示栏。

构造方法
GameView() 创建一个 GameView 的实例。其长宽、位置、背景等属性均为缺省设置。

类 litz.Button

继承自： litz.Div

此类定义了主菜单和游戏界面中用到的按钮的样式。在鼠标移入、点击按钮时，按钮会呈现出不同的样式。可以对按钮的点击事件绑定回调以实现点击按钮触发新事件的功能。

字段	
static Color	normalBackgroundColor 按钮的背景颜色。缺省值 Color(255,255,255,80)。
static Color	hoverBackgroundColor 鼠标悬停于按钮上时按钮的背景颜色。缺省值 Color(255,255,255,160)。
static Color	normalTextColor 按钮的文字颜色。缺省值 Color.black。
static Color	pressTextColor 鼠标按下时按钮的文字颜色。缺省值 Color.white。

构造方法

Button(String text, int posx, int posy, int width, int height)

创建一个具有指定文字、位置、长宽的按钮。

参数：

text – 按钮上的文字
posx – 按钮左上角的 x 坐标
posy – 按钮左上角的 y 坐标
height – 按钮的高，单位为像素
width – 按钮的宽，单位为像素

类 litz.Option

继承自： litz.Div

此类描述了一个选项栏（litz.OptionPanel 类的实例）中的一个选项。通过鼠标点击一个选项，可将选项栏中的其它选项置为未选中状态，并将点击的选项置为选中状态。被选中的选项有图标指示。鼠标移入和移出选项时，选项的背景色会相应地发生改变。

字段	
static Image	image_check 选项被选中时的指示图标使用的图片资源。
static Image	image_uncheck 选项未被选中时的指示图标使用的图片资源。
static ImageIcon	icon_check 选项被选中时的指示图标。
static ImageIcon	icon_uncheck 选项未被选中时的指示图标。
JLabel	checkbox 选项的指示图标模块。

构造方法

Option(String text, OptionPanel _optionPanel, int posx, int posy, int width, int height)

创建一个 Option 的实例，指定它的选项文字、属于哪一个选项栏，具有指定的坐标和长宽。

参数：

text - 选项的文字描述
_optionPanel - 选项隶属的选项栏
posx - 组件左上角的 x 坐标
posy - 组件左上角的 y 坐标
width - 组件的宽，单位为像素
height - 组件的高，单位为像素

类 litz.OptionPanel

继承自： litz.Div

此类定义了游戏规则选项界面的选项栏。在创建房间页面下用鼠标点击一个选项，可以向游戏内核发起修改游戏规则的申请。

字段	
int	selectedIndex 记录选项栏内被选中的选项的编号。选项从 0 开始编号。
int	height 选项栏组件的高度，单位为像素。
int	nOptions 选项栏内的选项个数。
litz.Option[]	options 保存选项栏内的选项的引用的数组。

构造方法

OptionPanel(int posx, int posy, int width, int _height)

创建一个 OptionPanel 的实例，具有指定的坐标和长宽。

参数：

posx – 组件左上角的 x 坐标

posy – 组件左上角的 y 坐标

width – 组件的宽，单位为像素

_height – 组件的高，单位为像素

方法	
void	selectOption(int index) 将选项栏内被选中的选项改为第 index 个选项。选项从 0 开始标号。 **参数：** index – 被选中的选项的标号

类 litz.MainMenu

继承自： litz.Div

此类实现了主菜单界面。主菜单界面包括开始画面、创建房间页面、输入邀请码页面和加入房间页面。详见“图 x 主菜单界面的菜单层级关系”。

字段	
boolean	optionEditable 控制游戏规则模块是否可被用户修改。若用户处于创建房间页面则值为 true，用户处于加入房间页面则值为 false。
litz.OptionPanel	optionPanel0 棋盘大小的选项栏。
litz.OptionPanel	optionPanel1 目标连子数的选项栏。
litz.OptionPanel	optionPanel2 是否设置禁手的选项栏。
litz.OptionPanel	optionPanel3 胜利条件的选项栏（传统/进阶）。传统胜利条件和进阶胜利条件的定义见？？？
litz.OptionPanel	optionPanel0 棋盘大小的选项栏。
Div	optionArea 规则设置模块。
Div	playerArea 房间内玩家列表模块。
litz.ButtonList	menu0 开始画面菜单
Div	menu1 创建房间/输入验证/加入房间页面。通过禁用和隐藏页面上的部分模块实现不同的页面。
litz.ButtonList	menu2 创建房间页面游戏控制菜单
litz.ButtonList	menu3 加入房间页面返回菜单
litz.TextField	invitationCodeField 创建房间页面显示邀请码以及客户端输入邀请码的文本框。
litz.Button	buttonJoin 加入房间页面加入房间的按钮。
RoomPlayerList	roomPlayerList 创建房间/加入房间页面内的玩家列表。

构造方法
MainMenu() 创建一个 MainMenu 页面，并初始化主菜单界面下的所有页面以及页面上的模块和菜单。所有页面在初始化时均被加入到 MainMenu 下，通过控制每个页面的显示与否来控制用户当前所在的页面。

方法	
void	paintComponent(Graphics g) 定义本类实例的绘制方法，绘制画面右上方的游戏标题。不应该被主动调用。 **参数：**

	g – 此组件的图形上下文

类 litz.Animator

继承自： Object

此类的静态方法用于发起对一个 swing 组件的渐慢移动动画，并在此类的一个实例中存储此次动画的目标位置。当对一个组件发起多个时间有重叠的动画时，计算这些动画的叠加，更新目标位置，并发起更新后的动画。

字段	
int	aimx 动画的目标位置的 x 坐标。
int	aimy 动画的目标位置的 y 坐标。

构造方法

private Animator(int x, int y)

创建一个具有指定目标位置的渐慢动画。无法被主动调用。

参数：

x – 动画的目标位置的 x 坐标。

y – 动画的目标位置的 y 坐标。

方法	
static void	animate(JComponent com, int dx, int dy) 给定相对于当前位置的移动量，对组件 com 发起一个新的渐慢动画。如果该组件正在动画过程中，将此次动画的位移与上次动画的目标位置叠加，终止上次动画，并发起叠加目标位置后的新动画。 **参数：** com – 应用此次动画效果的组件 dx – 此次动画 x 轴方向的相对移动量 dy – 此次动画 y 轴方向的相对移动量

类 litz.TaskTranslationGrav

继承自： TimerTask

此类实现了控制组件渐慢移动的线程。每一帧此线程会根据该组件绑定的动画计算组件的坐标的改变量并予以更新。

字段	
static double	dt 定义每两帧间的时间间隔。缺省为 16.67ms，即每秒 60 帧。

构造方法
TaskTranslationGrav(JComponent _com, Animator _ani) 指定一个组件和控制该组件的动画，启动一个控制该组件渐慢移动的线程。 **参数：** _com - 受本线程控制的组件 _ani - 控制该组件的动画

方法	
void	run() 定义线程每一帧的工作，修改组件的坐标。不应该被主动调用

类 litz.Primitive

继承自： Object

此类描述了动态背景中浮动的图元。一个图元可以定义成任意形状的多边形，在程序实现中，一个图元被定义成一个正方形。所有图元在背景中自左向右移动，绕自身中心顺时针缓慢旋转。

字段	
int[]	vx 记录图元的所有顶点的 x 坐标。
int[]	vy 记录图元的所有顶点的 y 坐标。
int	n 记录图元的顶点数量。
int	diag 记录正方形图元中心到顶点的长度。单位为像素。
Color	col 记录图元的颜色。
double	posx 图元的中心的 x 坐标。
double	posy 图元的中心的 y 坐标。
double	vel_translation 图元的移动速度。单位为像素/帧。
double	vel_rotation 图元的旋转速度。单位为弧度/帧。
double	theta 记录图元从生成到当前帧旋转的总弧度。

构造方法

Primitive(String type, int _diag)

创建一个指定类型的 Primitive 的实例。程序实现中，图元的类型始终为正方形，_diag 表示正方形中心到一个顶点的距离，单位为像素。

参数：

type - 图元的类型，程序实现中值始终为"Rect"

_diag - 正方形图元中心到一个顶点的距离，单位为像素

方法	
Color	generateColor() 在一定范围内随机生成一个颜色。一定范围内保证了随机生成的颜色与背景色 Color(0x778899) 色调相近。 **返回：** Color 类的实例，表示一个颜色。

类 litz.TaskAnimateBackground

继承自： TimerTask

此类实现了控制动态背景自动刷新的线程。每一帧此线程会计算每个图元的坐标和转角的改变量并予以更新，在必要时增删在背景中的图元。

构造方法
TaskAnimateBackground(AnimatedBackground _bg) 指定一个动态背景组件，启动一个控制该动态背景的线程。 **参数：** _bg - 受本线程控制的动态背景组件

方法	
void	run() 定义线程每一帧的工作，包括修改图元属性、增删图元。不应该被主动调用

类 litz.AnimatedBackground

继承自： litz.Div

此类实现了主页面和游戏界面的动态背景。动态背景包括纯色的背景和若干从左到右缓慢移动和绕自身中心顺时针缓慢旋转的 Primitive 类的实例，即图元。一个图元在完全移动到背景组件外时会被删除；新的图元会在背景组件的左侧不断生成。

字段	
HashSet<Primitive>	S 用于保存动态背景内的每一个图元。

构造方法

AnimatedBackground(int posx, int posy, int height, int width, Color col)

创建一个 AnimatedBackground，具有指定的坐标、长宽和背景颜色。

参数：

posx – 组件左上角的 x 坐标
posy – 组件左上角的 y 坐标
height – 组件的高，单位为像素
width – 组件的宽，单位为像素
col – 组件的背景颜色

方法	
protected void	paintComponent(Graphics g) 定义本类实例的绘制方法。不应该被主动调用。 **参数：** g – 此组件的图形上下文

类 litz.Main

继承自： Object

此类定义了游戏启动时需要进行的初始化行为，包括创建窗口、初始化背景组件并添加到窗口中、载入图片和字体资源等。此类的静态方法提供了初始化游戏界面、初始化主菜单、发起落子请求的功能。一个游戏进程中只会有一个 Main 类的实例。

字段	
static JFrame	appWindow 游戏窗口。
static litz.AnimatedBackground	rootPanel 主菜单界面和游戏界面的背景组件。此组件被直接添加到游戏窗口 appWindow 内，并且游戏中所有其余组件都添加到以 rootPanel 为根的组件树中。
static String[]	playerName 保存游戏中玩家的昵称的列表。
static int	nPlayer 保存游戏中玩家人数
static int	boardSize 保存游戏中棋盘的长宽。游戏中创建的棋盘长和宽是相等的。
static int	windowHeight 游戏窗口的高度，单位为像素。缺省为 768。
static int	windowWidth 游戏窗口的宽度，单位为像素。缺省为 1024。
static litz.ChessBoard	chessboard 保存游戏中的棋盘的引用。
static litz.ButtonList	playerList 保存游戏中的玩家列表的引用。
static HashMap <JComponent,Animator>	animatorMap 记录游戏中组件和正在控制该组件的动画间的对应关系，即 JComponent 类及其子类的实例到 Animator 类的实例的映射。
static Font	font_twcenmt 游戏中文字使用的字体的其中一种。
static Color	color_transparent_0 完全透明的颜色常量。
static Color	color_transparent_60 75%透明白色的颜色常量。
static litz.MainMenu	mainMenu 保存游戏中主菜单界面的引用
static litz.GameView	gameView 保存游戏界面的引用。
static Color	backgroundColor 游戏中动态背景的背景颜色。缺省为 Color(0x778899)。
static litz.GUI	gui 为游戏图形界面和游戏内核提供通信接口的类的实例。

构造方法
Main()

创建一个 Main 的实例。创建 Main 的实例后游戏即开始初始化，包括创建窗口、初始化背景组件并添加到窗口中、载入图片和字体资源。

方法	
static void	initializeGameView(JPanel rootPanel, int _nPlayer, int _boardSize, String[] _playerName) 初始化游戏界面，并添加到背景组件 rootPanel 下。需要传入玩家人数、棋盘大小和玩家昵称这些与界面外观相关的参数。 **参数：** rootPanel – 背景组件的引用 _nPlayer – 本局游戏的玩家人数 _boardSize – 棋盘的行列数 _playerName – 玩家昵称的数组
static void	initializeMainMenu(JPanel rootPanel) 初始化主菜单界面，并添加到背景组件 rootPanel 下。 **参数：** rootPanel – 背景组件的引用
static void	requestPutPawn(int x, int y, int player) 发起当前玩家在指定位置落子的请求到游戏内核。 **参数：** x – 落子位置的行号（从 0 开始标号） y – 落子位置的列号（从 0 开始标号） player – 当前玩家编号（从 1 开始标号）