**package edu.nju.main(主函数)：**

JmineSweeper：main方法，初始化MainFrame，各种游戏逻辑层操作处理单元，并让游戏界面实例观察游戏逻辑层操作处理单元，初始化OperationQueue并启动该线程，开始游戏。

**package edu.nju.controller.msgqueue(玩家操作顺连续执行的线程包)：**

OperationQueue：继承了Runnable，是一个玩家操作队列，当玩家进行了操作，操作会按先后顺序添加到其中，线程会不断取出并将其执行，若操作全部取出或者操作数已满，则会进入阻塞阶段，直至有新的操作。

**package edu.nju.controller.service(游戏控制层接口包，该包中接口主要在controller.impl实现)：**

ClientControllerService：作为客户端建立网络连接的接口，提供建立客户端的待执行方法。

GameControllerService：是一个接口，提供了游戏左击右击双击的待实行方法。

HostControllerService：作为主机建立网络连接的接口，提供建立主机的待执行方法。

MenuControllerService：是一个接口，提供了开始游戏的待执行方法。

SettingControllerService：是一个接口，提供了设置游戏难度的方法(包括自定义游戏)。

**package edu.nju.controller.impl(游戏控制层操作，主要由view.listener中的类调用，该包主要连接controller.service的接口，调用controller.msgqueue.operation包的类)：**

ClientControllerImpl：连接了ClientControllerService接口，当点击client后调用，建立客户端有主机连接。

GameControllerImpl：连接了GameControllerService接口，当发生游戏操作时建立对应的MineOperation，并将其添加到OperationQueue的队列中。

HostControllerImpl：连接了HostControllerService接口，当点击host后调用，建立主机等待客户端连接。

MenuControllerImpl：连接了MenuControllerService接口，当发生菜单操作时建立对应的MineOperation，并将其添加到OperationQueue的队列中。

SettingControllerImpl：连接了SettingControllerService接口，当发生菜单设置游戏难度操作时建立对应的MineOperation，并将其添加到OperationQueue的队列中。

**package edu.nju.controller.msgqueue.operation(玩家操作实现的连接包，该包中类主要在controller.impl中调用)：**

MineOperation：抽象类，是玩家操作的父类，含有抽象方法execute，需要子类根据各自的操作进行实例化。

LeftClickOperation：继承自MineOperation，左单击操作，会得到操作对应的方格，并通过ChessBoardModelService将其在逻辑层挖开。

DoubleClickOperation：继承自MineOperation，左双击操作，会得到操作对应的方格，并通过ChessBoardModelService将其在逻辑层迅速挖开。

RightClickOperation：继承自MineOperation，右单击操作，会得到操作对应的方格，并通过ChessBoardModelService将其在逻辑层标记。

StartGameOperation：继承自MineOperation，开始游戏操作，会通过GameModelService根据gameLevel设置大中小3中游戏难度，并开始游戏。

**package edu.nju.model.service(游戏逻辑层接口包，该包中接口主要在model.impl实现)：**

ChessBoardModelService：是一个接口，内含有初始化棋盘，设置棋盘宽高雷数，挖去，标记，快速挖开的待实现方法。

GameModelService：一个接口类，在游戏逻辑层上控制整体游戏状态，有设置棋盘尺寸，设置gameLevel，开始游戏，结束游戏的待实现方法。

ParameterModelService：一个接口类，在游戏逻辑层操作游戏雷数，有设置雷数，加减雷数的待实现方法。

StatisticModelService：一个接口类，游戏记录相关方法，有在界面上显示统计结果，记录游戏结果并在界面上显示等待实现方法。

**package edu.nju.model.impl(游戏逻辑层操作，部分类被view包的部分类观察，该包主要连接model.service的接口)：**

BaseModel：基本逻辑单元，继承了Obersevable，内含有更新方法updateChange，调用此方法会通知所有观察其的UpdateMessage类型的对象。

ChessBoardModelImpl：继承BaseModel，连接了ChessBoardModelService，被MineBoardPanel观察，具体在逻辑上实现初始化棋盘，设置棋盘宽高雷数，挖去，标记，快速挖开的方法，每次改变后会调用BaseModel的updateChange方法，并会根据棋盘的状态返回它在显示时应该显示的状态。

GameLevel：各个游戏难度信息，并将level或name与棋盘大小以及雷数对应 ，当设置等级时会从GameLevel的链表中取出对应的游戏等级。

GameModelImpl：继承BaseModel，连接了GameModelService，被MainFrame观察，初始化了三种游戏等级，功能玩家选择选区具体难度，具体实现了设置棋盘尺寸，设置gameLevel，开始游戏，结束游戏方法，并可转化成GameVO。

ParameterModelImpl：继承BaseModel，连接了ParameterModelService，被MineNumberLabel观察，推测：实现游戏上方雷数记录的加减操作，每次改变后会调用BaseModel的updateChange方法，并返回它显示时应该显示的状态。

StatisticModelImpl：继承BaseModel，连接了StatisticModelService，推测：实现游戏记录和保存游戏记录的操作，每次改变后会调用BaseModel的updateChange方法，并返回它显示时应该显示的状态。

UpdateMessage：游戏更新信息，构造时会得到操作类型，游戏棋盘状态。

**package edu.nju.model.po(游戏逻辑层信息)：**

BlockPO：逻辑层上每个扫雷块的信息，含有该块坐标，是否为雷，并可将扫雷块逻辑信息转化成显示信息。

StatisticPO：游戏层上游戏记录信息，包括游戏等级，总盘数，胜率。

**package edu.nju.model.vo(游戏显示层信息)：**

BlockVO：显示层上的扫雷块信息，包括扫雷块坐标以及当前状态。

GameVO：显示层上游戏信息，包括棋盘宽高，游戏等级，游戏一局状态，游戏结果状态，一局游戏时间

**package edu.nju.model.data(游戏记录数据)：**

StatisticData：获取游戏记录，保存游戏记录。

**package edu.nju.model.state(所有类是枚举，各种游戏状态)：**

BlockState：枚举，逻辑层上扫雷块状态：UNCLICK(未点击)，CLICK(点击)，FLAG(标记)。

DisplayBlockState：枚举，显示层上扫雷块状态：ZERO(附近0个雷)，ONE(附近1个雷)，TWO(附近2个雷)，THREE(附近3个雷)，FOUR(附近4个雷)，FIVE(附近5个雷)，SIX(附近6个雷)，SEVEN(附近7个雷)，EIGHT(附近8个雷)，UNCLICK(未点击)，FLAG(标记)， MINE(未点击但是雷)，Bomb(正在点击且是雷)，ERROFLAG(已标记但不是雷)；并可根据扫雷块附近雷数返回对应的符号。

GameResultState：枚举，游戏结果状态：SUCCESS(胜利)，FAIL(失败)，INTERRUPT(中途结束)。

GameState：枚举，游戏一局状态：RUN(运行)，PAUSE(暂停)，OVER(结束)。

**package edu.nju.view(游戏界面相关类，部分类对model.impl包中的部分类进行观察，从而根据游戏逻辑层变化进行相应显示变化)：**

CustomDialog：游戏自定义界面，可以设计棋盘宽高以及雷数，并可以返回游戏信息。

Images：游戏所有图片，含有根据参数不同返回不同ImageIcon的不同方法。

Location：扫雷块的坐标信息。

MainFrame：连接了Observer的接口，观察GameModelImpl，游戏主界面，初始化游戏窗口，棋盘面板，菜单栏，并加入窗口监听(在关闭时需要保存数据)，并可根据在界面层游戏信息改变时重新开始游戏或者结束游戏。

MineBoardPanel：继承JPanel，连接了Obserber的接口，观察ChessBoardModelImpl，棋盘面板，初始化所有扫雷快，并给所有扫雷块加入CoreListener监听，同时还会根据界面层扫雷块变化进行对应扫雷块的重绘。

MineNumberLabel：继承JLabel，连接了Observer的接口，观察ParameterModelImpl，左上角剩余雷数标签，会根据已标记雷的个数不断刷新。

MyButton：继承JLable，扫雷块按钮，含有paint以及paintComponent方法进行按钮的重绘。

RecordDialog：游戏记录界面，有显示，初始化的方法。

**package edu.nju.view.listener(游戏监听包，逻辑操作调用controller.impl包中的类的方法)：**

CoreListener：连接了MouserListener、ActionListener，用于监听玩家对游戏界面的操作(包括对棋盘，小人的点击操作)，并得到位置，根据操作类别(左单击，右单击，左双击)调用GameControllerService或MenuControllerService对应的方法。

MenuListener：连接了ActionListener，用于监听玩家对菜单栏的操作，根据点击按钮类别调用MenuControllerService或SettingControllerService或HostControllerService或ClientControllerService对应的方法。

**package edu.nju.network(网络连接时的基本数据与信息包)：**

Configure：提供了建立网络连接所需要的常量，例如IP、端口号、信息类型。

TransformObject：连接了Serializable的接口，序列化后用于在客户端与主机之间进行传递，构造函数中有String和UpdateMessage。

**package edu.nju.network.host(建立服务端以服务端读取和写出信息有关的所有操作的包)：**

ServerAdapter：建立host，并开启HostThread线程，同时拥有HostInHandlerImpl类，是服务端的连接类。

HostInHandler：一个接口，获得从客户端传来的数据，并将其加入到OperationQueue中等待执行。

HostInHandlerImpl：连接了HostInHandler接口，具体实现inputHandle(Object data)方法，将传来的数据转化成TransformObject并进行传递。

HostService：一个抽象类，连接了Observer类，是host服务类，可以让ServerAdapter创建服务器，同时用于向客户端端传送host的操作，含有初始化、结束、上传操作的方法(通过调用ServerAdapter的方法实现)。

HostServiceImpl：继承了HostService，具体实现上传操作的方法以及根据观察对象变化而被调用的更新方法。

HostThread：继承了Thread，线程不断等待读取客户器端传来的信息，并含有向客户器端写信息和关闭连接的具体实现方法。

传出数据：HostService观察到操作，通过ServerAdapter调用HostThread的方法传递给客户端。

接收数据：HostThread接收，通过ServerAdapter调用HostInHandler的方法将其加入OperationQueue中等待执行。

**package edu.nju.network.client(建立客户端以及客户端读取和写出信息有关的所有操作的包)：**

ClientAdapter：建立client，并开启ClientThread线程，同时拥有ClientInHandlerImpl类，是客户端的连接类。

ClientInHandler：一个接口，获得从服务器端传来的TransformObject数据，并将其中的UpdateMessage传给对应的Proxy。

ClientInHandlerImpl：继承了Observable类，连接了ClientInHandler，inputHandle(Object data)的具体实现。

ClientService：一个抽象类，是client服务类，可以让ClienAdapter创建客户端，同时用于向服务器端传送client的操作，含有初始化、结束、上传操作的方法(通过调用ClienAdapter的方法实现)。

ClientServiceImpl：继承了ClientService，具体实现上传操作的方法。

ClientThread：继承了Thread，线程不断等待读取服务器端传来的信息，并含有向服务器端写信息和关闭连接的具体实现方法。

传出数据：modelProxy包中的类调用ClientService的submitOperation(MineOperation op)方法，通过ClientAdapter调用ClientThread的方法传递给服务端。

接收数据：ClientThread接收，通过ClientAdapter调用ClientInHandler的方法将其传递给观察它的modelProxy包中的类。

**package edu.nju.network.modelProxy(在网络联机中作为client的model.impl代理包)：**

ModelProxy：继承了BaseModel类(也就是说继承了Observable类)，连接Observer接口，是所有代理类的基类，用于在网络对战时代替client的model.impl包中的类。

GameModelProxy：继承了ModelProxy类，连接GameModelImpl接口，是client在网络对战中GameModelImpl的代理类。