# 详细设计文档

团队名称：六毛软糖

目录

[详细设计文档 1](#_Toc390029953)

[1. 引言 1](#_Toc390029954)

[2. 产品概述 1](#_Toc390029955)

[3. 结构视角 1](#_Toc390029956)

[4. 依赖视角 33](#_Toc390029957)

## 引言

##### 1.1编制目的

本报告详细完成对天天爱消除游戏软件系统的详细设计，达到指导后续软件构造的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

##### 1.2参考资料

《软件需求规格说明文档》，六毛软糖

《软件体系结构描述文档》，六毛软糖

## 产品概述

参考《项目说明文档》；

## 结构视角

##### 3.1 User模块设计：

##### 3.1.1 user模块的设计如图一：

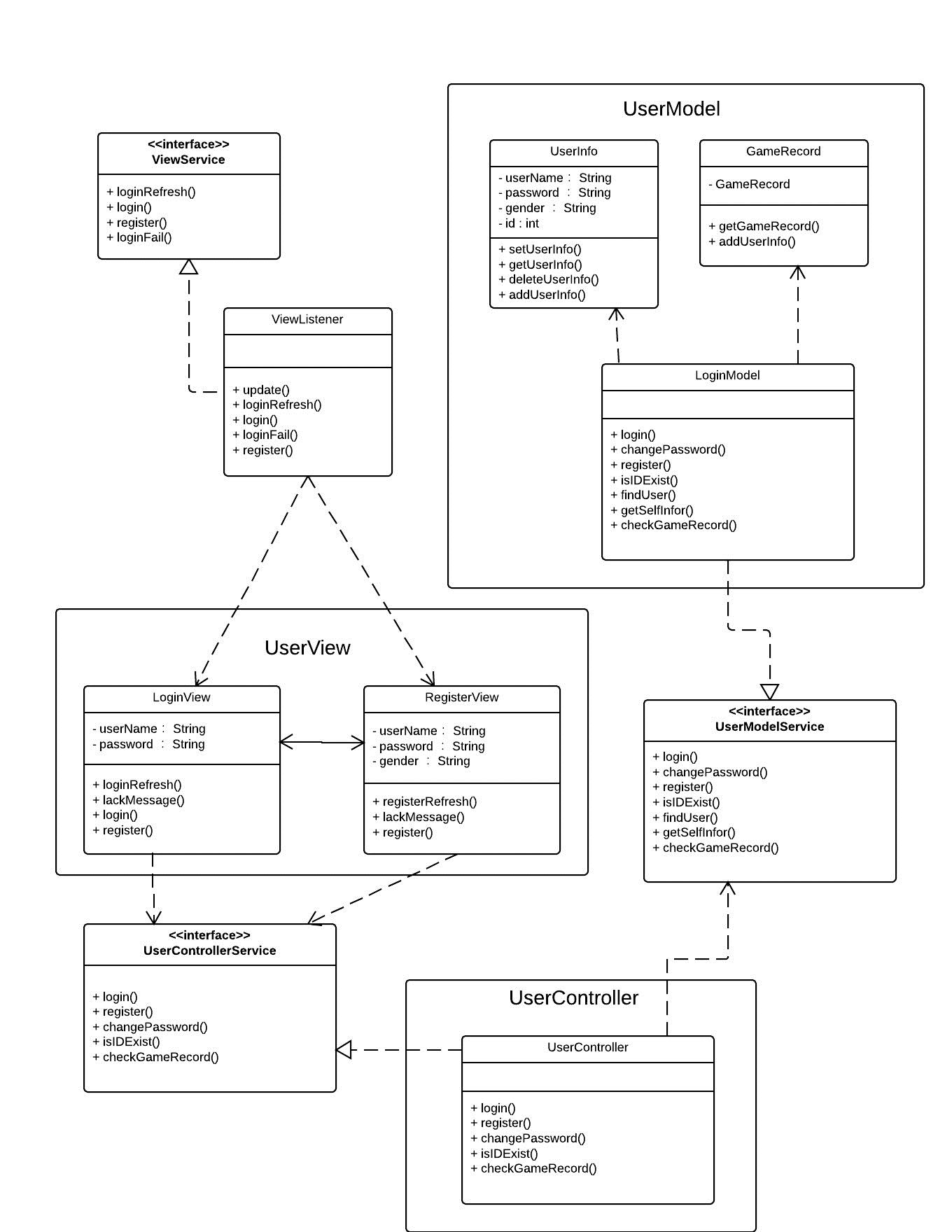


图 一

##### 3.1.2 user模块各个类的职责如表：

|  |  |
| --- | --- |
| 类名 | 职责 |
| Message | 处理View层传来的发送和管理好友申请，游戏邀请等 |
| User | 处理View层传来的对于用户各种属性信息查看的需求 |

###### 3.1.3 userView的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserUI.loginRefresh | 语法 | Private void loginRefresh() |
| 前置条件 | UserModel模块返回注册成功 |
| 后置条件 | 由注册界面跳转到登录界面 |
| UserUI.loginFail | 语法 | Private void loginFail() |
| 前置条件 | UserModel模块返回登录失败 |
|  | 界面提示登录失败，请重新登录 |
| RegisterUI.refresh | 语法 | Private void refresh() |
| 前置条件 | 用户选择注册功能 |
| 后置条件 | 跳转到注册界面 |
| RegisterUI.fail | 语法 | Private void fail() |
| 前置条件 | UserModel模块返回用户已存在 |
|  | 界面提示用户已存在，请重新填写注册信息 |
| 需要的服务（需接口） | | |

|  |  |  |
| --- | --- | --- |
| UserController.login | 语法 | public userPO login(String userName,String password) |
| 前置条件 | password符合输入规则 |
| 后置条件 | 查找是否存在相应的user，根据输入的password返回登录验证的结果 |
| UserController.register | 语法 | public boolean register(String userName,String password,String gender) |
| 前置条件 | 各项信息符合输入规则 |
| 后置条件 | 查找是否有重复，返回添加结果 |
| UserController.  changePassword | 语法 | public String changePassword(String userName,String newPassword) |
| 前置条件 | 输入的密码符合规则 |
| 后置条件 | 查找是否有相应的User，返回密码修改结果 |
| UserController.  checkGameRecord | 语法 | Public GameRecord checkGameRecord(String id) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 返回查找结果 |
| MainUI.refresh | 语法 | public void refresh() |
| 前置条件 | 用户登录成功 |
| 后置条件 | 界面跳转到主界面 |

###### 3.1.4userControl的接口规范

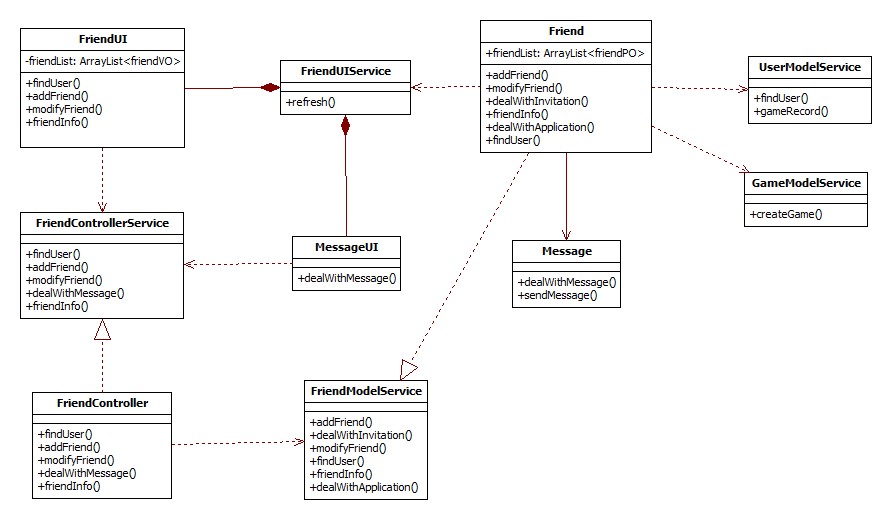
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserController.login | 语法 | public void login(String id,String password) |
| 前置条件 | password符合输入规则 |
| 后置条件 | 查找是否存在相应的user，根据输入的password返回登录验证的结果 |
| UserController.register | 语法 | Public String register(String userName,String password,boolean gender,int image) |
| 前置条件 | 各项信息符合输入规则 |
| 后置条件 | 查找是否有重复，返回结果信息 |
| UserController.  changePassword | 语法 | Public String changePassword(String userId,String newPassword) |
| 前置条件 | 输入的密码符合规则 |
| 后置条件 | 查找是否有相应的User，返回密码修改结果信息 |
| UserController.logoff | 语法 | public String logoff(String id) |
| 前置条件 | 用户选择关闭窗口或者退出 |
| 后置条件 | 系统将用户设置为离线状态 |
| UserController.  checkGameRecord | 语法 | Public GameRecordVO checkGameRecord(String id) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 返回查找结果 |
| UserController.  sendApplication | 语法 | Public void sendApplication(ApplicationVO avo) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 系统将申请发送到服务器端 |
| UserController.  sendApplication | 语法 | Public void sendApplication(ApplicationVO avo) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 系统将申请发送到服务器端 |
| UserController.  sendInvitation | 语法 | Public void sendInvitation(InvitationVO ivo) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 系统将申请发送到服务器端 |
| UserController.  replyApplication | 语法 | public void replyApplication(ApplicationVO aVo,boolean reply) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 系统将回复发送到服务器端 |
| UserController.  replyInvitation | 语法 | public void replyInvitation(InvitationVO iVo,boolean reply) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 系统将回复发送到服务器端 |
| 需要的服务（需接口） | | |
| UserModel.login | 语法 | public String login(String userName,String newPassword) |
| 前置条件 | 输入的密码符合规则 |
| 后置条件 | 查找是否有相应的User，返回密码修改结果 |
| UserModel.  changePassword | 语法 | public String changePassword(String userName,String newPassword) |
| 前置条件 | 输入的密码符合规则 |
| 后置条件 | 查找是否有相应的User，返回密码修改结果 |
| UserModel.register | 语法 | public boolean register(String userName,String password,String gender) |
| 前置条件 | 各项信息符合输入规则 |
| 后置条件 | 查找是否有重复，返回添加结果 |
| UserModel.isIDExist | 语法 | public boolean isIDExist(String userName) |
| 前置条件 | 无 |
| 后置条件 | 返回查找结果 |
| UserModel.getSelfInfor | 语法 | public UserPO getSelfInfor(String id) |
| 前置条件 | 用户已成功登录 |
| 后置条件 | 返回该用户的个人信息 |
| UserModel.  checkGameRecord | 语法 | public GameRecord checkGameRecord(String id) |
| 前置条件 | 用户已成功登录 |
| 后置条件 | 返回该用户的游戏记录 |

###### 3.1.5userModel对外接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| UserModel.register | 语法 | public String register(UserPO up） |
| 前置条件 | 无 |
| 后置条件 | 返回注册结果 |
| UserModel.changePassword | 语法 | public boolean changePassword(String userId, String newPassword) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 返回修改结果信息 |
| UserModel.login | 语法 | public String login(String userId, String password) |
| 前置条件 | 无 |
| 后置条件 | 返回结果信息 |
| UserModel.logoff | 语法 | public void logoff(String id) |
| 前置条件 | 用户已经登录 |
| 后置条件 | 系统更改用户状态信息 |
| UserModel.  checkGameRecord | 语法 | public GameRecordPO checkRecord(String id) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 返回查找结果 |
| UserModel.getMessage | 语法 | public ArrayList<MessagePO> getMessage(String id) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 系统将申请发送到服务器端 |
| UserModel.saveMessage | 语法 | public void saveMessage (MessagePO message); |
| 前置条件 | 用户登录成功 |
| 后置条件 | 系统将申请发送到服务器端 |
| UserModel.getInvitations | 语法 | public ArrayList<MessagePO> getInvitations(String id) ; |
| 前置条件 | 用户登录成功 |
| 后置条件 | 系统将申请发送到服务器端 |
| UserModel.deleteInvitations | 语法 | public void deleteInvitations(String id); |
| 前置条件 | 用户登录成功 |
| 后置条件 | 系统将回复发送到服务器端 |
| UserController.  replyInvitation | 语法 | public void replyInvitation(InvitationVO iVo,boolean reply) |
| 前置条件 | 用户登录成功 |
| 后置条件 | 系统将回复发送到服务器端 |

##### 3.2 Friend 模块设计：

###### 3.2.1 Friend模块的设计类图



###### 3.2.2 FriendView的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FriendUI.refresh | 语法 | Public void refreshFriend（ArrayList<FriendVO> friendList） |
| 前置条件 | 用户选择了与好友管理有关的操作 |
| 后置条件 | 系统保存该操作效果，并更新好友列表 |
| MainUI.refresh | 语法 | Public void refreshFriend（ArrayList<FriendVO> friendList） |
| 前置条件 | 用户回到系统主界面，即排行榜界面 |
| 后置条件 | 系统显示当前该玩家排行榜 |

|  |  |  |
| --- | --- | --- |
| 需要的服务（需接口） | | |
| FriendController.addFriend | 语法 | Public void addFriend(UserVO uv) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 无 |
| FriendController.findUser | 语法 | Public UserVO findUser(String id) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 无 |
| FriendController.dealWithMessage | 语法 | Public void dealWithMessage(String answer) |
| 前置条件 | 用户成功登录并收到好友消息 |
| 后置条件 | 无 |
| FriendController.modifyFriend | 语法 | Public void modifyFriend（FriendVO fv） |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |
| FriendController.friendInfo | 语法 | Public RecordVO friendInfo（string friendId） |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |

###### 3.2.3 FriendControl的接口规范

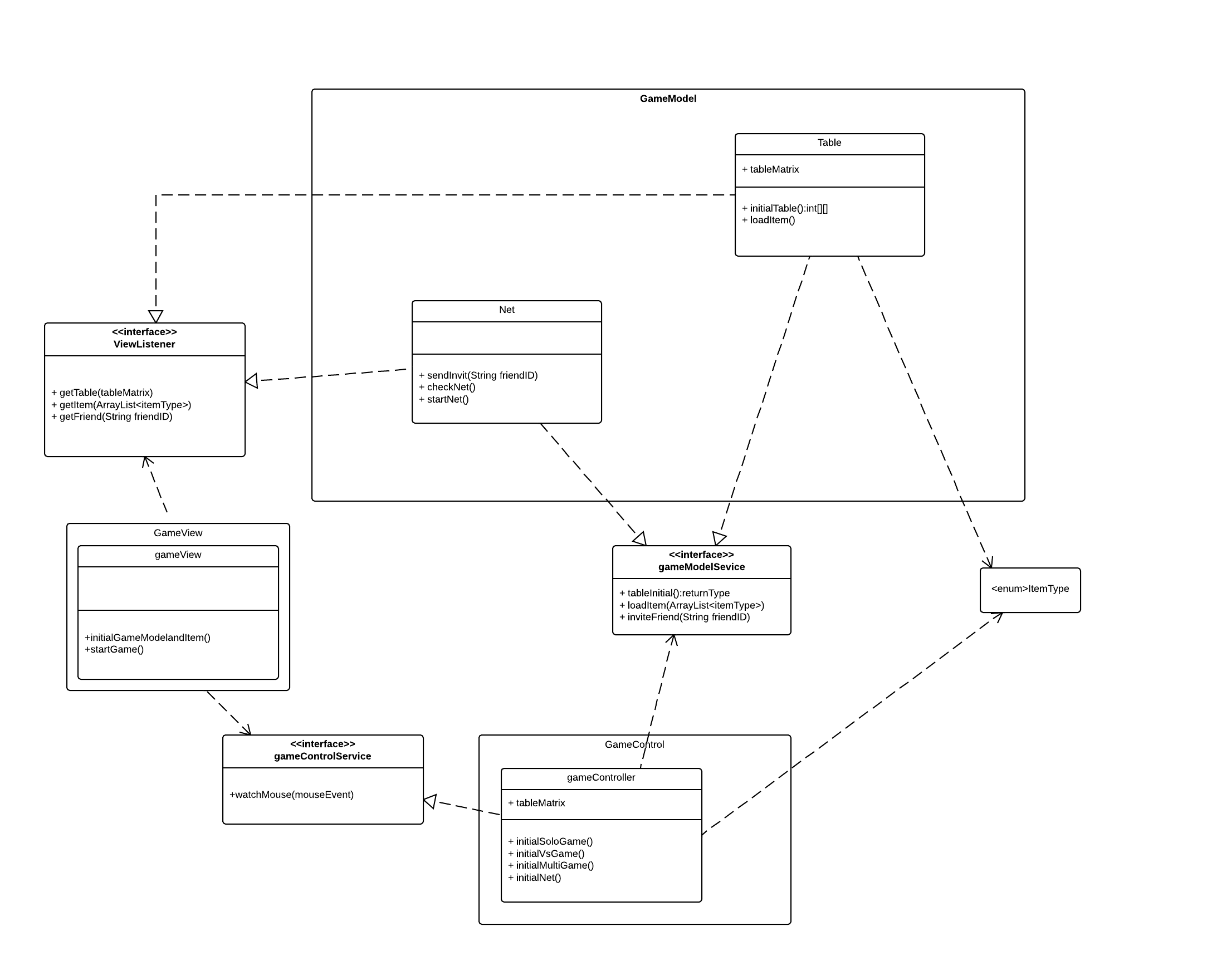
|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FriendController.addFriend | 语法 | Public void addFriend(UserVO uv) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 无 |
| FriendController.findUser | 语法 | Public UserVO findUser(String id) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 无 |
| FriendController.dealWithMessage | 语法 | Public void dealWithMessage(String answer) |
| 前置条件 | 用户成功登录并收到好友消息 |
| 后置条件 | 无 |
| FriendController.modifyFriend | 语法 | Public void modifyFriend（FriendVO fv） |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |
| 需要的服务（需接口） | | |
| FriendModel.addFriend | 语法 | Public void addFriend(UserPO, up,FriendPO fp) |
| 前置条件 | 用户成功登录并请求添加好友 |
| 后置条件 | 无 |
| UserModel.findUser | 语法 | Public UserPO findUser（UserPO up） |
| 前置条件 | 用户成功登录并查找用户 |
| 后置条件 | 无 |
| GameModel.createGame | 语法 | Public void createGame（） |
| 前置条件 | 用户成功登录并接受游戏邀请 |
| 后置条件 | 无 |
| FriendModel.modifyFriend | 语法 | Public void modifyFriend（FriendPO fp） |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |
| FriendModel.sendMessage | 语法 | Public void sendMessage(message) |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |

###### 3.2.4 FriendModel的接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务（供接口） | | |
| FriendModel.addFriend | 语法 | Public void addFriend(FriendPO fp) |
| 前置条件 | 用户成功登录并请求添加好友 |
| 后置条件 | 无 |
| UserModel.findUser | 语法 | Public UserPO findUser（UserPO up） |
| 前置条件 | 用户成功登录并查找用户 |
| 后置条件 | 无 |
| UserModel.gameRecord | 语法 | Public recordPO gameRecord(string userId) |
| 前置条件 | 用户成功登录 |
| 后置条件 | 无 |
| GameModel.createGame | 语法 | Public void createGame（） |
| 前置条件 | 用户成功登录并接受游戏邀请 |
| 后置条件 | 无 |
| FriendModel.modifyFriend | 语法 | Public void modifyFriend（FriendPO fp） |
| 前置条件 | 用户成功登录并拥有一个以上好友 |
| 后置条件 | 无 |
| 需要的服务（需接口） | | |
| FriendUI.refresh | 语法 | Public void refreshFriend（ArrayList<FriendVO> friendList） |
| 前置条件 | 用户选择了与好友管理有关的操作 |
| 后置条件 | 系统保存该操作效果，并更新好友列表 |
| MainUI.refresh | 语法 | Public void refreshFriend（ArrayList<FriendVO> friendList） |
| 前置条件 | 用户回到系统主界面，即排行榜界面 |
| 后置条件 | 系统显示当前该玩家排行榜 |

##### 3.3 Game 模块设计

###### 3.3.1Game模块的设计类图



###### 3.3.2 GameVIew的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务接口（供接口） | | | | |
| GameUI.initialTable | 语法 | | Public void initialTable() | |
| 前置条件 | | 用户单击开始本局游戏 | |
| 后置条件 | | 显示初始化的棋盘，初始化棋盘中不允许已有超过三个以上可消除的图案 | |
| GameUI.selectGameMode | 语法 | | Public void selectGameMode(int gameMode) | |
| 前置条件 | | 用户单击开始游戏 | |
| 后置条件 | | 用户选择本局游戏使用的道具 | |
| GameUI. selectItem | 语法 | | Private void selectItem(int itemNum) | |
| 前置条件 | | 用户已选择游戏模式 | |
| 后置条件 | | 系统初始化棋盘，跳转到游戏界面 | |
| 需要的服务（需接口） | | | | |
| GameController.creatGame | | 语法 | | Public void creatGame(Game g) |
| 前置条件 | | 用户选择完游戏模式、道具并单击开始游戏 |
| 后置条件 | | 显示游戏界面 |

###### 3.3.3 GameControl内部类的职责

|  |  |  |
| --- | --- | --- |
| 类名 | 方法 | 职责 |
| gameControl | initialSoloGame() | 初始化单机游戏 |
| initialFriendsNet() | 作为游戏发起者初始化协作游戏并进行网络连接 |
| initialMultiGameAsParticipate() | 作为游戏参与者初始化协作游戏并进行网络连接 |
| initialRivalsandFriendsNet() | 初始化对战游戏并进行网络连接 |

###### 3.3.4 GameControl的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务（供接口） | | | | |
| GameController.creatGame | 语法 | | Public void creatGame(Game g) | |
| 前置条件 | | 用户选择完游戏模式、道具并单击开始游戏 | |
| 后置条件 | | 显示游戏界面 | |
| 需要的服务（需接口） | | | | |
| GameModel.creatTableModel | | 语法 | | Public void creatTableModel() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 创建棋盘初始矩阵 |
| GameModel.loadItem | | 语法 | | Public void loadItem() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 加载本局可使用的道具 |
| GameModel.sentGameInvitation | | 语法 | | Public void sentGameInvitation() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 发送游戏邀请 |
| GameModel.sentGame | | 语法 | | Public void sentGame(String IP) |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 发送游戏参数 |

###### 3.3.5 GameModel 的内部类职责

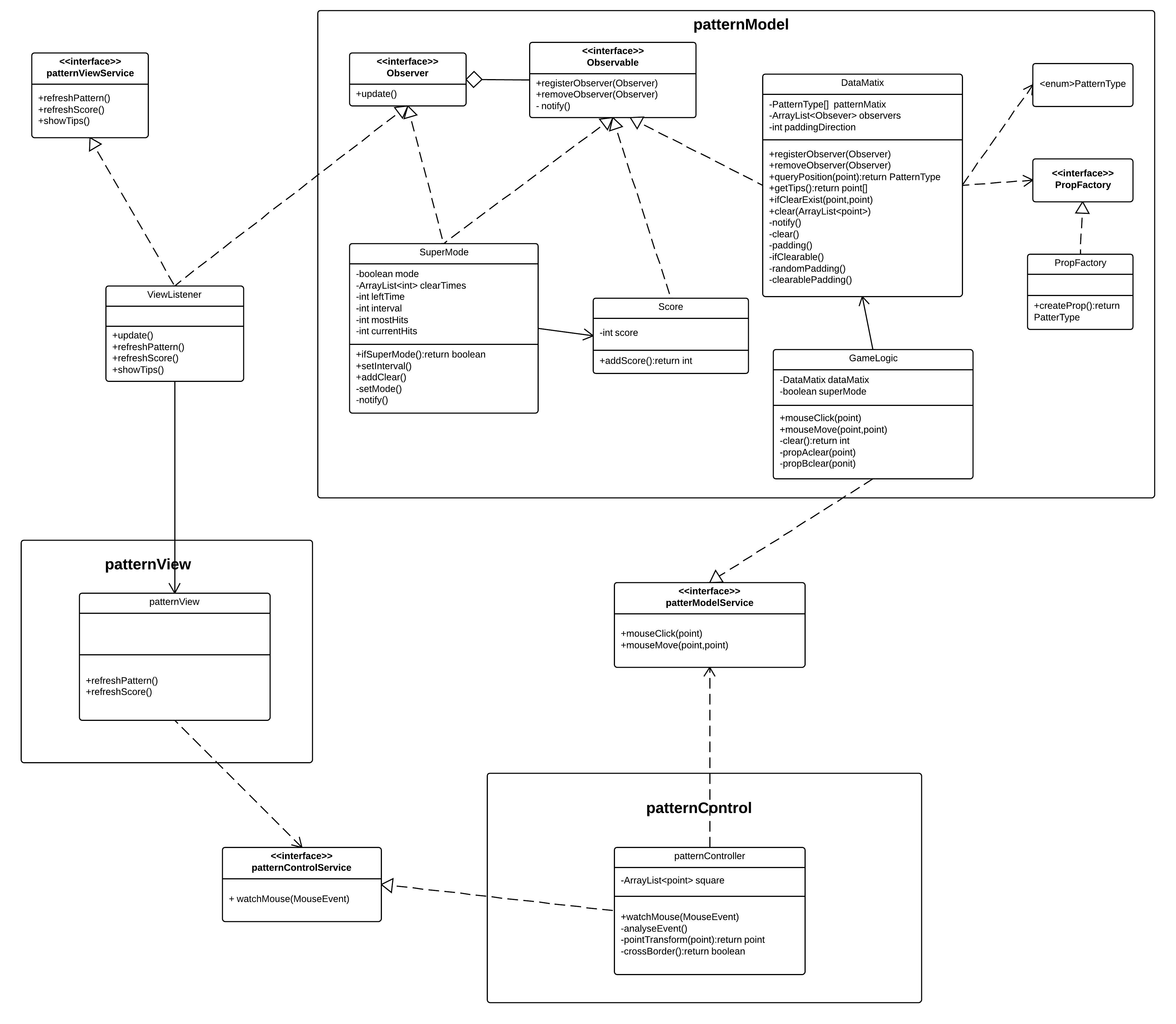
|  |  |  |
| --- | --- | --- |
| 类名 | 方法 | 职责 |
| GameState | getInstance() | 得到保存游戏数据的对象 |
| set() | 设置游戏逻辑对象，监听以及道具的选择 |
| startGame() | 开始进行游戏数据记录 |
| endGame() | 结束游戏数据记录 |
| ifSuperMode() | 返回超级模式的状态 |
| haltGame() | 暂停游戏数据记录 |
| restartGame() | 重新开启游戏数据记录 |
| update() | 当有消除操作时，提供给被监听者调用的接口 |
| registerObserver() | 注册对自己的监听 |
| removeObserver() | 移除对自己的监听 |
| notifyViewObserver() | 通知界面的监听者刷新界面 |
| addRival() | 设置对战游戏中的对手 |
| broadcastGameStart() | 通知对战游戏中的对手游戏开始 |
| getHit() | 返回本局游戏的最大连击数 |
|  | getScore() | 返回本局游戏的分数 |
|  | getGameDate() | 返回本局游戏的开始时间 |

###### 3.3.6 GameModel的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务（供接口） | | | | |
| GameModel.creatTableModel | | 语法 | | Public void creatTableModel() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 创建棋盘初始矩阵 |
| GameModel.loadItem | | 语法 | | Public void loadItem() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 加载本局可使用的道具 |
| GameModel.sentGameInvitation | | 语法 | | Public void sentGameInvitation() |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 发送游戏邀请 |
| GameModel.sentGame | | 语法 | | Public void sentGame(String IP) |
| 前置条件 | | 控制模块创建一局游戏 |
| 后置条件 | | 发送游戏参数 |
| 需要的服务（需接口） | | | | |
| GameUI.initialTable | 语法 | | Public void initialTable() | |
| 前置条件 | | 用户单击开始本局游戏 | |
| 后置条件 | | 显示初始化的棋盘，初始化棋盘中不允许已有超过三个以上可消除的图案 | |
| GameUI.  selectGameMode | 语法 | | Public void selectGameMode(int gameMode) | |
| 前置条件 | | 用户单击开始游戏 | |
| 后置条件 | | 用户选择本局游戏使用的道具 | |
| GameUI. selectItem | 语法 | | Private void selectItem(int itemNum) | |
| 前置条件 | | 用户已选择游戏模式 | |
| 后置条件 | | 系统初始化棋盘，跳转到游戏界面 | |

##### 3.4 Pattern模块设计

###### 3.4.1 Pattern模块的设计类图



###### 3.4.2 patternView的接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PatternUI.  refreshPattern | 语法 | | Public void refreshPattern(ArrayList<patternVO> patternsList) | |
| 前置条件 | | 用户的单击对界面产生影响 | |
| 后置条件 | | 无 | |
| PatternUI.  RefreshScore | 语法 | | Public void refreshScore(int score) | |
| 前置条件 | | 用户的游戏积分发生改变 | |
| 后置条件 | | 无 | |
| PatternModel.  propsA | 语法 | | Private void propsA(point p) | |
| 前置条件 | | 用户单击的坐标是一个道具A | |
| 后置条件 | | 消除周围8个图案 | |
| patternUI.showTips | 语法 | | Private void patternUI.showTips (point p) | |
| 前置条件 | | 界面生成消除提示 | |
| 后置条件 | | 无 | |
| 需要的服务（需接口） | | | | |
| PatternController.  watchMouse | | 语法 | | Public void watchMouse(MouseEvent mouseEvent) |
| 前置条件 | | 游戏已初始化，用户在游戏界面有鼠标操作 |
| 后置条件 | | 根据不同的鼠标事件调用相应的逻辑操作 |

###### 3.4.3 Pattern Control内部类职责

|  |  |  |
| --- | --- | --- |
| 类名 | 方法 | 职责 |
| patternControl | mouseClick() | 用于接受界面的鼠标单击行为并操作游戏数据 |
| mouseDrag() | 用于接受界面的鼠标拖动行为并操作游戏数据 |
| startGame() | 开始游戏 |
| startGameWithRivals() | 在对战游戏中与对手一同开始游戏 |
| haltGame() | 暂停游戏 |
| restartGame() | 暂停游戏后继续游戏 |
| saveGameRecord() | 游戏结束后保存游戏数据 |
| propClear | generateClears() | 根据道具类型生成相应的消除坐标 |
| RemoteControl | run() | 执行命令 |
| setCommand() | 设置命令 |
| getReturn() | 得到命令执行结果 |
| GetGameRecordCommand | execute() | 执行获取游戏数据命令 |
| HaltGameCommand | execute() | 执行暂停游戏命令 |
| RestartGameCommand | execute() | 执行继续游戏命令 |
| PropClearCommand | execute() | 执行道具消除命令 |
| QueryCommand | execute() | 执行查询位置命令 |
| StartGameCommand | execute() | 执行开始游戏命令 |
| StartGamePassivelyCommand | execute() | 执行作为游戏的参与者被动地开始游戏以实现同步命令 |
| SwapCommand | execute() | 执行交换两个图案位置命令 |
| WeakenPropCommand | execute() | 执行在对战游戏中削弱道具C效果的命令 |
| FrozenPatternCommand | execute() | 执行在对战游戏中冻结一种图案的命令 |
| NoCommand | execute() | 执行空白命令 |

###### 3.4.4 patternControl的对外接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务（供接口） | | | | |
| PatternController.  watchMouse | 语法 | | Public void watchMouse(MouseEvent mouseEvent) | |
| 前置条件 | | 游戏已初始化，用户在游戏界面有鼠标操作 | |
| 后置条件 | | 根据不同的鼠标事件调用相应的逻辑操作 | |
| 需要的服务（需接口） | | | | |
| PatternModel.  mouseClick | | 语法 | | Public void mouseClick(point cilckPoint) |
| 前置条件 | | 用户在游戏界面的鼠标操作为单击 |
| 后置条件 | | 根据单击的坐标判断后续逻辑操作 |
| PatternModel.  mouseMove | | 语法 | | Public void moveMouse(point p1,point p2) |
| 前置条件 | | 用户在游戏界面的鼠标操作为拖动鼠标 |
| 后置条件 | | 根据鼠标拖动的前后坐标判断后续逻辑操作 |
| PatternModel.showTips | | 语法 | | Public void showTips() |
| 前置条件 | | 用户2s内无任何操作 |
| 后置条件 | | 调用PatternView.showTips() |

###### 3.4.5 patternmodel的内部类职责

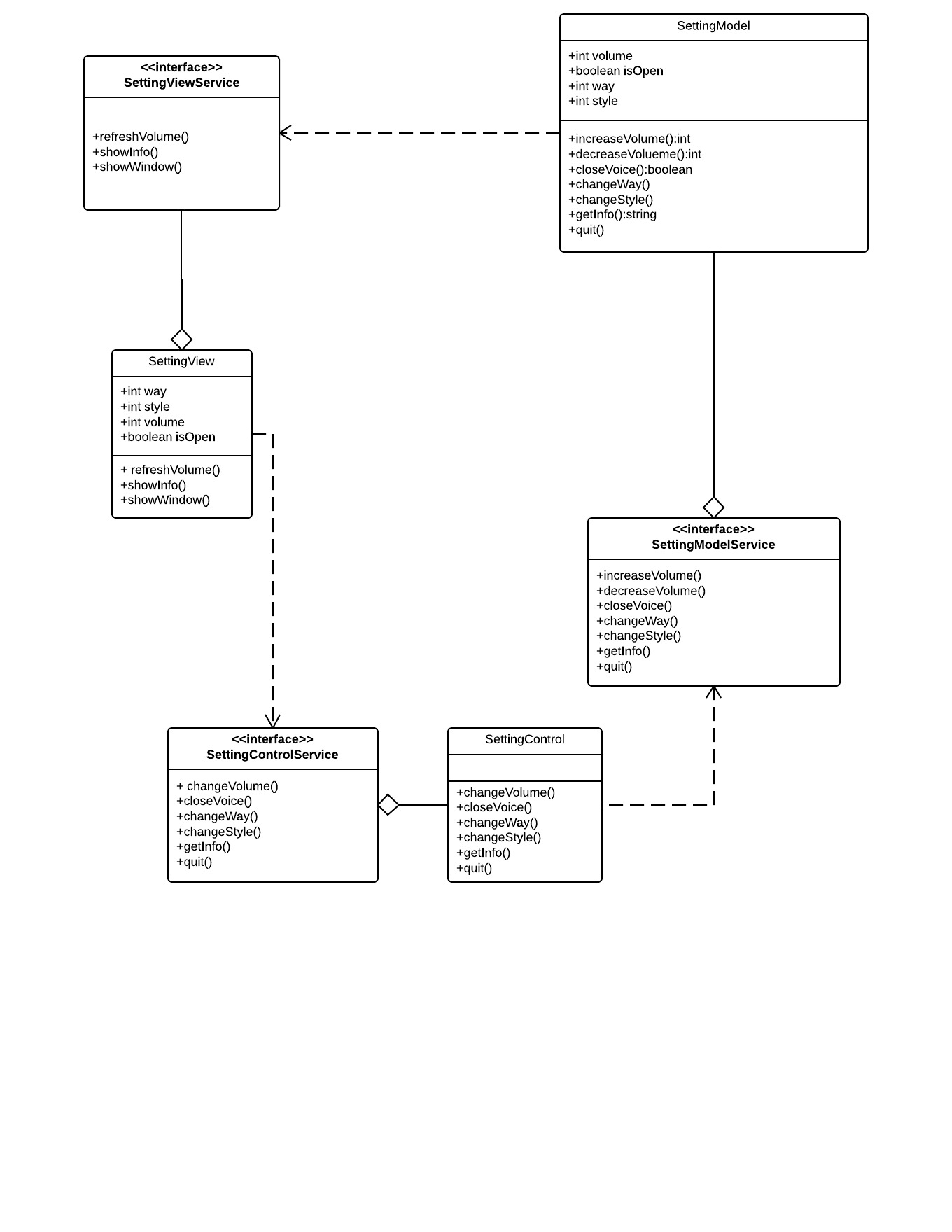
|  |  |  |
| --- | --- | --- |
| 类 | 方法 | 职责 |
| checkClearStragety.CheckifClearStragety | ArrayList<MatricLocation> checkClearList(ArrayList<MatricLocation> list) | 检查是否消除的策略的接口 |
| checkClearStragety.CheckClearStragety | ArrayList<MatricLocation> checkClearList(ArrayList<MatricLocation> list) | 检查是否消除的策略的具体策略类 |
| paddingStragety.PaddingStragety | ArrayList<MatricLocation> producePaddingList(ArrayList<MatricLocation> list) | 填充消除后的底层数据的接口 |
| paddingStragety.downPaddingStragety | 同上 | 实现了paddingStragety的向下掉落的具体策略类 |
| paddingStragety.RightPaddingStragety | 同上 | 实现了paddingStragety的向左掉落的具体策略类 |
| propStragety.PropStragety | MatricLocation produceProps(ArrayList<MatricLocation> list) | 生成道具的策略的接口 |
| propStragety.TwoPropStragety | 同上 | 实现了PropStragety的具体策略类 |
| tipsStragety.TipStragety | ArrayList<MatricLocation> getTips() | 给出可消除移动位置的策略类接口 |
| tipsStragety.TwoTipStragety | 同上 | 实现了PropStragety的策略类，生成道具A和道具B |
| DataMatric | public static DataMatric getDataMatric() | 获得DataMatric单间的静态方法 |
| public int getLength() | 获得底层数据数组的长度 |
| public int getDataMatricValue(MatricLocation location) | 获得一个位置上的底层数据的值 |
| public boolean setValue(MatricLocation x) | 设置一个位置上的底层数据的值 |
| public boolean isInLength(int value) | 判断是否超过了底层数据数组的长度 |
| DataMatricReaderHelper | public ArrayList<MatricLocation> checkClearList(ArrayList<MatricLocation> list) | 判断传入的list中的底层数据中是否是可消除的，返回可消除的位置 |
| public boolean isClearedWhenInit(MatricLocation location) | 初始化底层数据时使用，判断是否形成了消除 |
| public ViewType getViewTypeInDataMatric(MatricLocation x) | 获取DataMatric底层的数据类型，转换成ViewType使用,使用空的MatricLocation |
| public int getDataMatricLength() | 获得dataMatric的数组的长度 |
| DataMatricWriterHelper | public boolean changeValueInDataMatric(MatricValueLocation x,MatricValueLocation y) | 交换两个点的值，然后通知数据改变啦。 |
| public boolean setValueList(Collection<MatricLocation> list) | 设置底层数据的值 |
| GameLogic | public void addRivals(ArrayList<VsGameService> vs) | 添加对手 |
| public void setFrozenState() | 对外调用的方法，冻结对一定颜色的图案无法移动，使用线程， |
| public synchronized boolean changePatternLocation(MatricLocation location1,  MatricLocation location2) | 接受交换的命令，交换两个图案的位置 |
| public PatternType queryPosition(Point x) | 查询底层的位置下的种类，图案，或者道具,千万注意，当遇到Prop2时，会调用frozenComponent方法 |
| public void clear(ArrayList<Point> collection) | 消除一组图案 |
| public void registerObserver(Model.patternModel.Observer o) | 监听者模式 |
| public void removeObserver(Model.patternModel.Observer o) |
| public void notifyObserver() |
| MatricLocation | Getter and setter | 封装位置和值 |
| ViewType | public PatternType changeToPatternType(ViewType type) | 关于五种图案和两种道具的枚举类 |
| public static ViewType getTypeFromValue(int value) |

###### 3.4.6 patternmodel 的对外接口规范

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 提供的服务（供接口） | | | | |
| PatternModel.  mouseClick | | 语法 | | Public void mouseClick(point cilckPoint) |
| 前置条件 | | 用户在游戏界面的鼠标操作为单击 |
| 后置条件 | | 根据单击的坐标判断后续逻辑操作 |
| PatternModel.  moveMouse | | 语法 | | Public void moveMouse(point p1,point p2) |
| 前置条件 | | 用户在游戏界面的鼠标操作为拖动鼠标 |
| 后置条件 | | 根据鼠标拖动的前后坐标判断后续逻辑操作 |
| 需要的服务（需接口） | | | | |
| PatternUI.  refreshPattern | 语法 | | Public void refreshPattern(ArrayList<patternVO> patternsList) | |
| 前置条件 | | 用户的单击对界面产生影响 | |
| 后置条件 | | 无 | |
| PatternUI.  RefreshScore | 语法 | | Public void refreshScore(int score) | |
| 前置条件 | | 用户的游戏积分发生改变 | |
| 后置条件 | | 无 | |
| patternUI.showTips | 语法 | | Private void patternUI.showTips (point p) | |
| 前置条件 | | 界面生成消除提示 | |
| 后置条件 | | 无 | |

##### 3.5 setting模块设计

###### 3.5.1 setting模块的设计类图



###### 3.5.2 settingView的对外接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务接口（供接口） | | |
| SettingUI.refreshVolume | 语法 | Public void refreshVolume(int volume) |
| 前置条件 | 用户修改了游戏音量 |
| 后置条件 | 系统更新界面的音量数字显示 |
| SettingUI.showWindow | 语法 | Public void showWindow() |
| 前置条件 | 用户选择关闭/打开游戏音效 |
| 后置条件 | 系统弹出提示窗口 |
| SettingUI.showInfo | 语法 | Public void showInfo() |
| 前置条件 | 用户已经查看游戏相关信息 |
| 后置条件 | 系统显示出游戏相关信息的界面 |
| 需要的服务接口（需接口） | | |
| SettingController.increaseVolume(int volume) | 处理来自界面增大音量的输入 | |
| SettingController.decreaseVolume(int volume) | 处理来自界面减小音量的输入 | |
| SettingController.changeStyle(int style) | 处理来自界面选择的具体的游戏风格 | |
| SettingController.changeWay(int way) | 处理来自界面选择的具体的图案下落方式 | |

###### 3.5.3 settingControl的对外接口规范

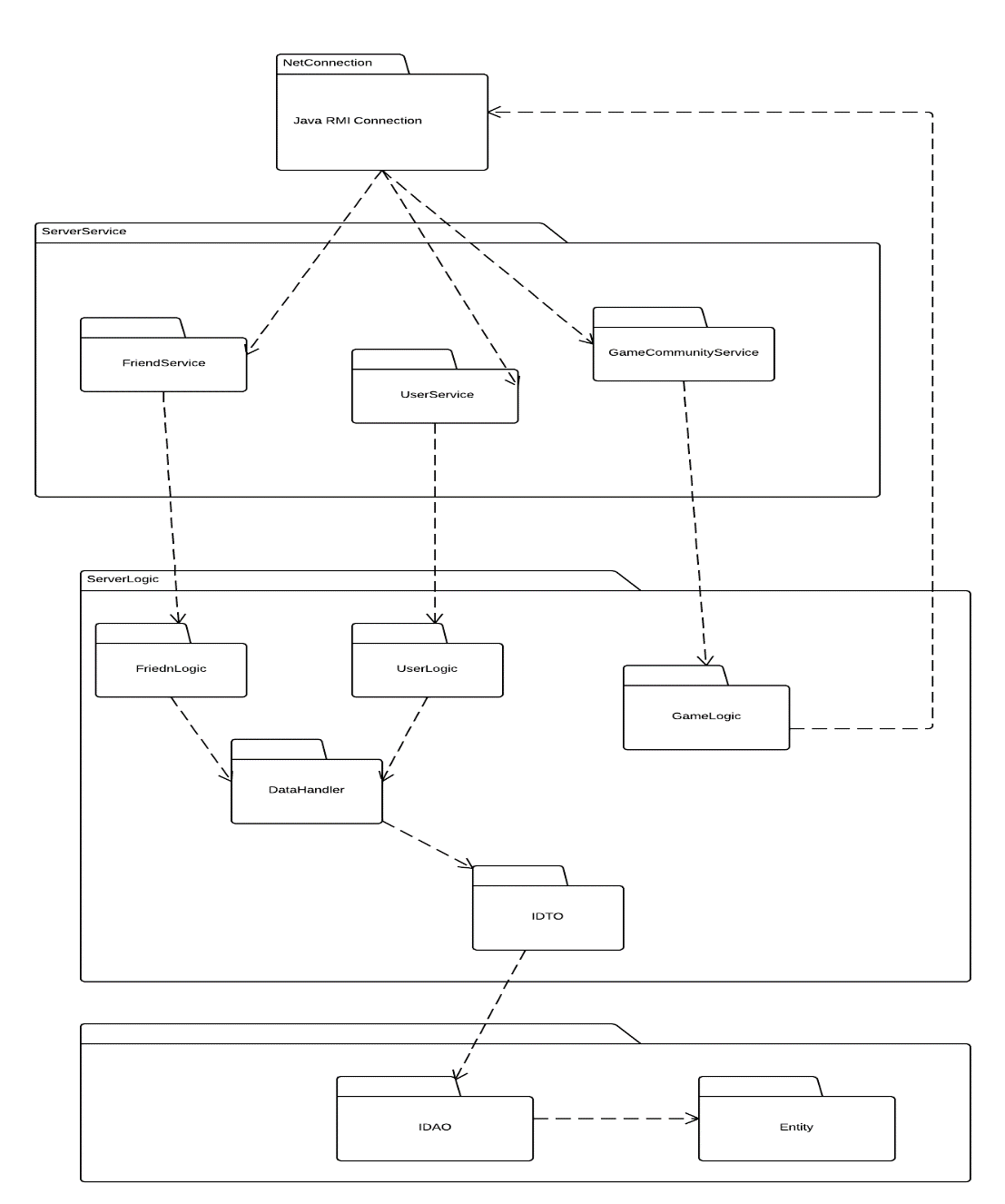
|  |  |  |
| --- | --- | --- |
| 提供的服务接口（供接口） | | |
| SettingController.increaseVolume | 语法 | Public void increaseVolume(MouseEvent mouseEvent,int volume) |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 根据不同的鼠标操作调用相应的逻辑操作 |
| SettingController.decreaseVolume | 语法 | Public void decreaseVolume(MouseEvent mouseEvent,int volume) |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 根据不同的鼠标操作调用相应的逻辑操作 |
| SettingController.changeStyle | 语法 | Public void changeStyle(MouseEvent mouseEvent,int style) |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 根据不同的鼠标操作调用相应的逻辑操作 |
| SettingController.changeWay | 语法 | Public void changeStyle(MouseEvent mouseEvent,int style) |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 根据不同的鼠标操作调用相应的逻辑操作 |
| SettingController.getInfo | 语法 | Public void getInfo(MouseEvent mouseEvent) |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 游戏信息内容不因用户的操作发生改变 |
| SettingController.quit | 语法 | Public void quit（MouseEvent mouseEvent） |
| 前置条件 | 游戏已经成功登录，用户进入游戏设置界面 |
| 后置条件 | 根据鼠标操作完成相应的逻辑操作 |
| 需要的服务接口（需接口） | | |
| SettingModel.increaseVolume  (int volume) | 增大游戏系统音量 | |
| SettingModel.decreaseVolume(int volume) | 减小游戏系统音量 | |
| SettingModel.closeVoice(boolean isOpen) | 打开或关闭游戏音效 | |
| SettingModel.changeStyle(int style) | 更改游戏图案风格 | |
| SettingModel.changeWay(int way) | 更改游戏图案下落方向 | |
| SettingModel.getInfo( ) | 获得游戏相关信息 | |
| SettingModel.quit( ) | 退出游戏设置 | |

###### 3.5.4 settingModel 的对外接口规范

|  |  |  |
| --- | --- | --- |
| 提供的服务接口（供接口） | | |
| SettingModel.increaseVolume | 语法 | Public void increaseVolume(int volumn) |
| 前置条件 | 用户选择增大音量 |
| 后置条件 | 系统增大游戏音量，并且调用SettingUI.refreshSetting |
| SettingModel.decreaseVolume | 语法 | Public void decreaseVolume(int volumn) |
| 前置条件 | 用户选择减小音量 |
| 后置条件 | 系统减小游戏音量，并且调用SettingUI.refreshSetting |
| SettingModel.closeVoice | 语法 | Public void closeVoice(boolean isOpen) |
| 前置条件 | 用户选择关闭/打开音效 |
| 后置条件 | 系统关闭/打开游戏音效，并且调用SettingUI.remindWindow |
| SettingModel.changeStyle | 语法 | Public void changeStyle(int style) |
| 前置条件 | 用户选择某种风格 |
| 后置条件 | 系统根据选择调换风格，内部修改风格属性 |
| SettingModel.changeWay | 语法 | Public void changeWay(int Way) |
| 前置条件 | 用户选择改变游戏下落方向 |
| 后置条件 | 系统根据选择的下落方向，内部进行游戏属性的修改 |
| SettingModel.getInfo | 语法 | Public void getInfo( ) |
| 前置条件 | 用户选择查看游戏相关信息 |
| 后置条件 | 系统调用游戏相关信息，并且调用SettingUI.showInfo |
| SettingModel.quit | 语法 | Public void quit( ) |
| 前置条件 | 用户放弃修改游戏设置 |
| 后置条件 | 系统离开游戏设置界面，并且调用gameUI.refresh |
| 需要的服务接口（需接口） | | |
| SettingUI.refreshVolume(int volume) | 更新游戏音量数字 | |
| SettingUI.showWindow( ) | 显示游戏音效更改的提示窗口 | |
| SettingUI.showInfo( ) | 显示游戏的相关信息窗口 | |
| mainUI.refresh | 刷新游戏主界面 | |

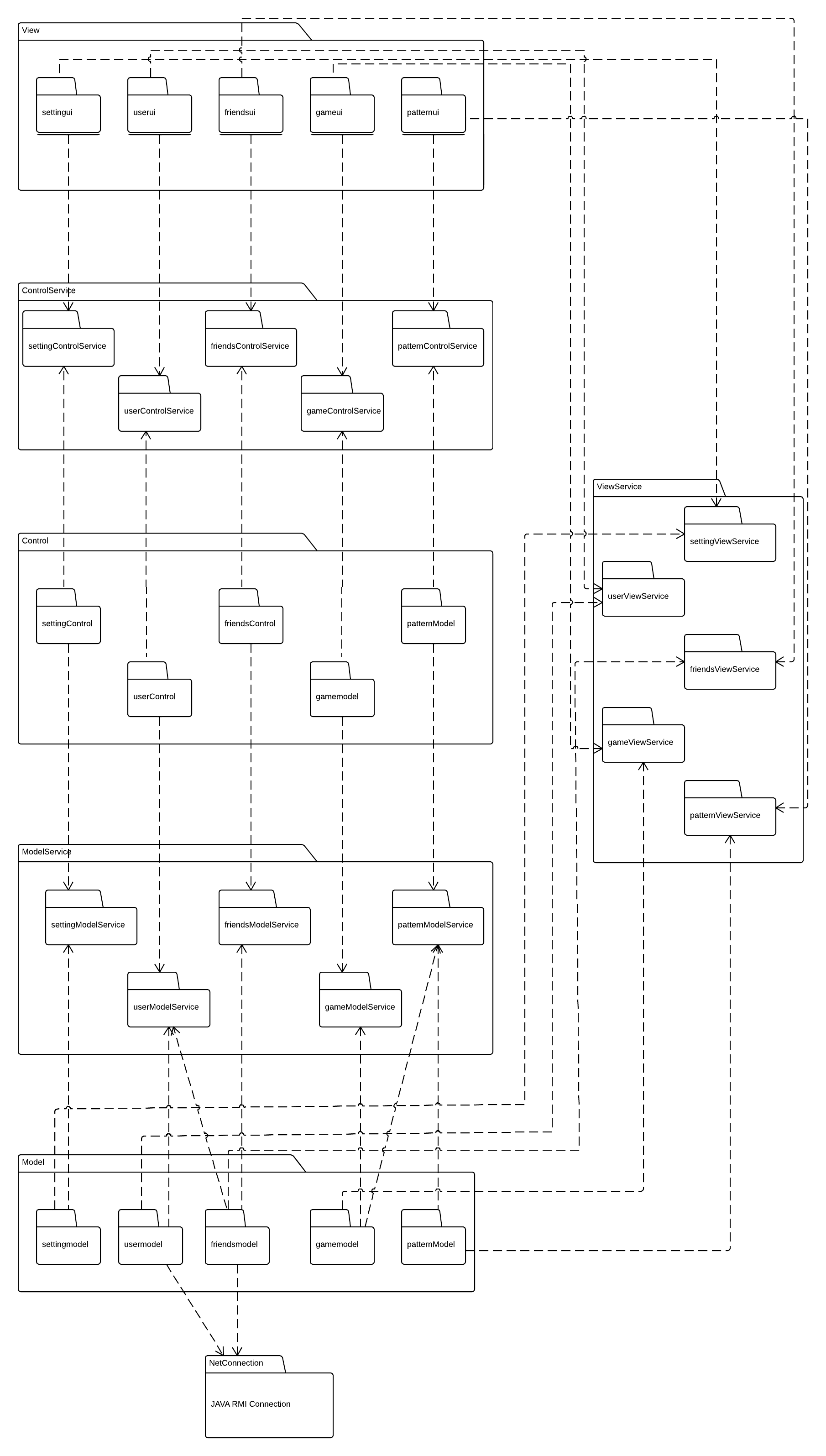
## 依赖视角

。



服务器开发包图

## 



客户端开发包图