AWARE_ness;

Not everything is as it seems...

Software Requirements Specification

JOHNS HOPKINS UNIVERSITY
FOUNDATIONS OF
SOFTWARE ENGINEERING
(EN.605.401.81.SU13)

Version 1.07

Date: 07/05/2013

PROJECT PLAN VERSION CONTROL

Version	Date	Author	Message
1.00	06/29/2013	Shan Sabri	Document and template initialization
1.01	07/01/2013	Shan Sabri	Initial Subsystem and Function structuring
1.01	07/01/2013	Hartanto Thio	Initial Subsystem and Function structuring
1.01	07/01/2013	Davis Gigogne	Initial Subsystem and Function structuring
1.02	07/03/2013	Shan Sabri	Modified project flow diagram to reflect in-game settings
1.03	07/04/2013	Davis Gigogne	UML Diagram addition
1.04	07/04/2013	Hartanto Thio	Use Case additions for the Player and Client subsystems
1.05	07/04/2013	Shan Sabri	Use Case additions for the Game subsystem
1.06	07/04/2013	Davis Gigogne	Use Case additions for the ClientManager subsystem
1.07	07/05/2013	Davis Gigogne	Interface communications added to Project Interface

INTRODUCTION

This document's purpose is to serve as the software requirements specification (SRS) document for Team Aware_ness' implementation of Project Clue-less' target system. The SRS will address the system overview in terms of subsystem architecture with corresponding domains, interfaces, and constraints.

TABLE OF CONTENTS

1	Version Control	1
2	Table of Contents	2
3	Project Scope	3
4	Project Flow	4
5	Project Features	5
6	Project Architecture	6 – 15
7	Project Interface	16
8	UML Diagram	17
9	Document Acceptance	18

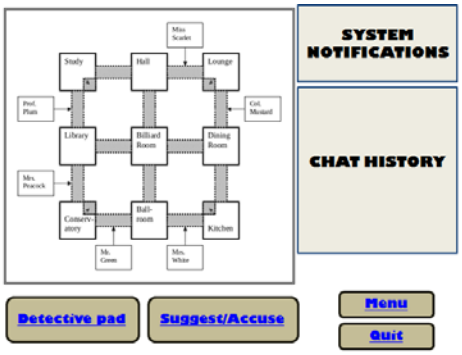
PROJECT SCOPE

Clue-less is a simplified implementation of the popular board game Clue©. Clue-less will include nine rooms, six weapons, and six characters. The main objective is to maneuver throughout the game board collecting clues from which to deduce which player murdered the game's victim with which weapon and in what room.

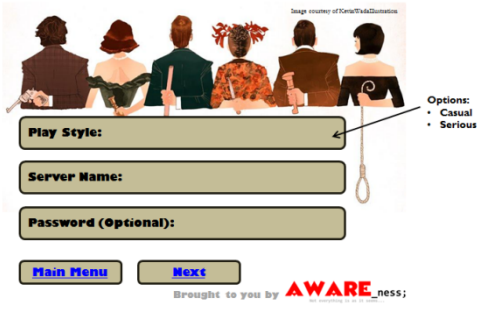
The goal of this project is to utilize a Java backend with a HTML frontend to efficiently play Clue-less with multiple users. The backend is intended to handle the logic and implementation need to play Clue-less, while the GUI front end will be used as a medium to communicate the user with the backend.

Below are a few useful screenshots to represent the program's capabilities:

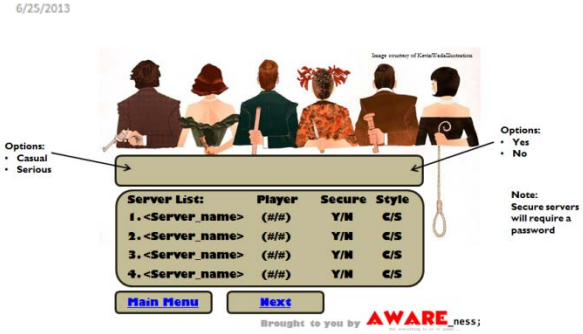
Main Menu



Game Screen



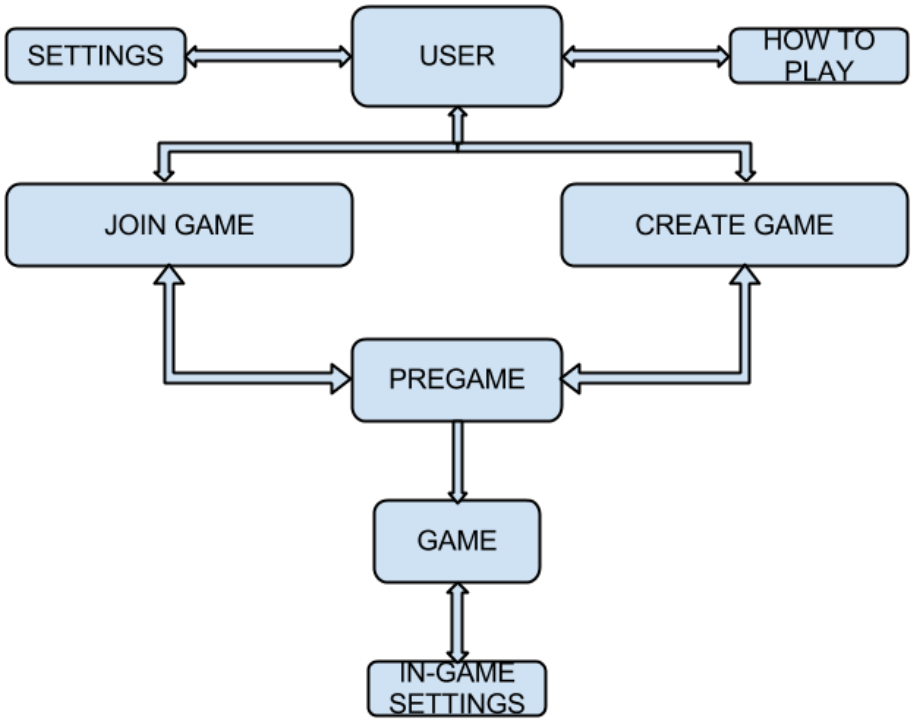
Create Game Menu



Join Game Menu

PROJECT FLOW

Below is a basic representation of Clue-less' workflow:



The user will have the ability to create a new game, join a previously created game, access the settings menu, or game rules. Each case requires specific protocol.

Case 1 (Create Game):

Assuming the user decides to create a new game, the user will be redirected to a pre-game settings menu where he/she will have the ability to name the server, secure the server, and label the server according to play style. Once the game has been created, the user will have the ability to play and communicate with users that have joined that particular game.

Case 2 (Join Game):

Assuming the user decides to join a previously created game, the user will be redirected to a pre-game screen where he/she will have the ability to filter available servers by play style (casual/serious) and security (yes/no). Upon selecting a server to join, the user will be directed to the corresponding game and be able to play and communicate with other users.

Case 3 (Settings):

Assuming the user decides to access the settings menu, the user will have the ability to adjust sound (on/off) and language (English, etc.) parameters.

Case 4 (Rules):

Assuming the user decides to access the game rules, the user will be redirected to Hasbro's official Clue game rules website via a new browser window.

PROJECT FEATURES

1. Cross-Platform support: users will have the ability to play Clue-less on all operating systems with a modern browser.

2. Language support: users will have the ability to selected from a list of language for global support

3. Chat: users within a game will have the ability to chat with one another.

4. System Notifications: users within a game will be notified of system messages and actions (e.g. play order)

5. Voting: users within a game will have the ability to vote to restart the game or to kick a player(s) from the game.

6. Ignore: users within a game will have the ability to ignore others from the chat log.

7. Filtering: users wishing to join a game will have the ability to filter servers by security and play style.

8. Muting: users will have the ability to mute system notification sounds

PROJECT ARCHITECTURE

Clue-less can be broken down into five main subsystems, each with specialized functions and limiting constraints.

1. GamesManager

The GamesManager class is required to handle any currently pending and active games.

a. CreateGame

Actor:	Player
Preconditions:	Player is currently not participating in a game
Postconditions:	Player has created a game
Flow:	Player must wait for at least two additional participants to begin game

b. JoinGame

Actor:	Player
Preconditions:	Player is currently not participating in a game and selected game contains a vacant position
Postconditions:	Player has joined a game
Flow:	Player can now play the game given at least three participants

c. DeleteGame

Actor:	System
Preconditions:	Game is currently not being played and no participants are on the sever
Postconditions:	Game has been deleted
Flow:	System updates query

d. QueryGame

Actor:	System
Preconditions:	Player filters available games to join by security and/or play style
Postconditions:	System updates query
Flow:	Player can now choose game based on a filtered selection

2. Game

The Game class is required to handle the players' conditions and game options.

a. Start

Actor:	System
Preconditions:	Players have entered a game
Postconditions:	Play order and solution will be determined
Flow:	Game has been initiated

b. AddPlayer

Actor:	System
Preconditions:	Player has selected a game to join and there is an available position
Postconditions:	Player has joined the pre-game screen
Flow:	Player will select a character based on availability

c. RemovePlayer

Actor:	All players
Preconditions:	ProcessVote has been initiated and majority of the players have decided to remove a player from the current game
Postconditions:	Player has been removed from the game
Flow:	Game will continue if there are at least three remaining participants. If not, the game will be terminated

d. NotifyAllPlayers

Actor:	System
Preconditions:	Game has been created and participants have entered the game
Postconditions:	System notifications will be displayed on the game screen for each player
Flow:	System will process player actions and notify all players

e. NotifyPlayer

Actor:	System
Preconditions:	The game has been created and participants have entered the game
Postconditions:	System notifications will display on the game screen for a particular player
Flow:	System will process a player's action and notify the corresponding player

f. ProcessAccusation

Actor:	System
Preconditions:	The game is active and the participants have been assigned clues
Postconditions:	Players have been notified of the accusation
Flow:	System updates and the game continues if the accusation is incorrect

g. ProcessSuggestion

Actor:	System
Preconditions:	The game is active and the participants have been assigned clues
Postconditions:	Players have been notified of the suggestion
Flow:	System updates and the game continues

h. ProcessEndTurn

Actor:	System
Preconditions:	It must be the current player's turn
Postconditions:	System will process the end of the current player's turn and moves on to the next player
Flow:	System updates and the game continues

i. ProcessQuit

Actor:	System
Preconditions:	The player must be in an active game
Postconditions:	The player is removed from the game
Flow:	Game will continue if there are at least three remaining participants. If not, the game will be terminated

j. ProcessVote

Actor:	System
Preconditions:	Players have voted by majority to restart the game or kick a player
Postconditions:	System will process the vote and notify the corresponding player(s)
Flow:	Game will either restart via RestartGame or a player will processed for removal via RemovePlayer

k. Restart

Actor:	All players
Preconditions:	ProcessVote has been initiated and majority of the players vote to restart the current game.
Postconditions:	Game will restart
Flow:	Game has been restarted and players have been reset with their corresponding parameters

3. Player

The Player class is required to allow the user to play the game.

a. Accuse

Actor:	Player
Preconditions:	There is an active game, it is the current player's turn, and the player has selected a character, weapon, and room for an accusation
Postconditions:	The system will determine the state of the accusation
Flow:	If the accusation is correct, the player wins the game If the accusation is incorrect, the player loses the game and can only DisproveSuggestion

b. Suggest

Actor:	Player
Preconditions:	There is an active game, it is the current player's turn, the player has selected a character, weapon, and room for a suggestion, and the player is located in a room
Postconditions:	The system will determine the state of the suggestion
Flow:	Suggestion has been made and current player may collect clues

c. DisproveSuggestion

Actor:	Player
Preconditions:	There is an active game and one of the players has already made a suggestion and the system has determined the suggested player based on their clues
Postconditions:	The user holding the selected clue will show the current player
Flow:	The system will update and the game continues

d. GetLanguage

Actor:	System
Preconditions:	The player has opened the game
Postconditions:	Returns a language that the user selected
Flow:	The system updates

e. SetLanguage

Actor:	System
Preconditions:	The player's preferred language exists within the game
Postconditions:	The system will update the player's preferred language
Flow:	The system updates

f. GetStatus

Actor:	System
Preconditions:	There is an active game with participants
Postconditions:	Returns if the players is active or inactive
Flow:	The system updates and the game continues

g. SetStatus

Actor:	System
Preconditions:	There is an active game with participants
Postconditions:	The system will update the player's status
Flow:	If the user makes an incorrect accusation or quits the game then the user is set as inactive, otherwise active

h. GetCharacter

Actor:	System
Preconditions:	There is an active game and the player is currently in the game
Postconditions:	Returns the character which the player is using
Flow:	The system updates and the game continues

i. SetCharacter

Actor:	System
Preconditions:	A game has been created or exists, the player's character has not been taken by other players, and the player is on the pre-game screen
Postconditions:	The system will update the player's character
Flow:	The system updates and continues to the game

j. GetClues

Actor:	System
Preconditions:	There is an active game and the player is currently in the game
Postconditions:	Returns a list of clues for that player
Flow:	The system updates and the game continues

k. SetClues

Actor:	System
Preconditions:	There is an active game and the player is currently in the game
Postconditions:	Assigns the player his/her clues given they are not already take
Flow:	The system updates and continues to the game

l. Vote

Actor:	Player
Preconditions:	The player is in an active game
Postconditions:	The system will process an majority vote before executing the action
Flow:	The player will have the ability to vote to restart or kick another player

m. Notify

Actor:	System
Preconditions:	The player is in an active game
Postconditions:	The player will be notified by the system of an event that has occurred
Flow:	The system will update and the notification will be pushed to the game screen

n. NotifyOthers

Actor:	Player
Preconditions:	The player is in an active game
Postconditions:	The player will send chat notifications to all unblocked players
Flow:	The player's message is displayed within the game screen to all unblocked players

o. EndTurn

Actor:	Player
Preconditions:	It must be current player's turn and the user chooses to not move, suggest, or accuse
Postconditions:	The player has ended his/her turn
Flow:	The game will continue with another player's turn

p. Quit

Actor:	Player
Preconditions:	The player is currently in a game
Postconditions:	The player is redirected to the main screen
Flow:	The player has exited the game

4. Client

The Client class is required to map the players' information to the client. Because this particular class is implemented through the front end, it is not reflected on the UML diagram listed on page 17.

a. MovePlayer

Actor:	Client
Preconditions:	The player must be in a current game and has selected or has been notified to move to a different position on the game board
Postconditions:	The client will animate the player's movement
Flow:	The game screen updates and the game continues

b. AccusePlayer

Actor:	Client
Preconditions:	It must be the current player's turn and the player has selected a character, weapon, and room to accuse
Postconditions:	The game will process the accusation and a corresponding prompt will display
Flow:	If the accusation is correct, the player will receive a win prompt If the accusation is incorrect, the player will receive a lose prompt

c. SuggestPlayer

Actor:	Client
Preconditions:	It must be the current player's turn and the player has selected a character, weapon, and room to suggest
Postconditions:	The game will process the suggestion and a corresponding prompt will display
Flow:	The player will see at most one of the clues being used in the disprove theory

d. AddNotification

Actor:	Client
Preconditions:	There is an active game with participants
Postconditions:	Depending on the notification, it will either be displayed to all player or a particular player
Flow:	The game screen updates and the game continues

e. AddMessage

Actor:	Client
Preconditions:	There is an active game with players and a player has chosen to send a message to the chat log
Postconditions:	The message will be added to the chat logs of the other unblocked users
Flow:	The chat log updates and the game continues

f. DisplayAccuseResponse

Actor:	Client
Preconditions:	There is an active game with players and one player has made an accusation
Postconditions:	If the player has at least one of the clues used in the accusation theory, he/she can use one clue to disprove that
Flow:	The client updates and the game continues if the accusation is incorrect

g. DisplaySuggestResponse

Actor:	Client
Preconditions:	There is an active game with players and one player has made a suggestion
Postconditions:	If the player has at least one of the clues used in the suggestion theory, he/she can use one clue to disprove that
Flow:	The client updates and the game continues

h. DisplayDetectivePad

Actor:	Client
Preconditions:	There is an active game with participants
Postconditions:	The detective pad will be displayed
Flow:	The player's notes and detective pad can be updated

i. DiplayInGameSettings

Actor:	Client
Preconditions:	There is an active game with players and a player is on the game screen
Postconditions:	The in-game settings screen will be displayed
Flow:	The player will have the ability to modify in-game settings

j. DisplaySettings

Actor:	Client
Preconditions:	The player is on the main game screen
Postconditions:	The game settings screen will be displayed
Flow:	The player will have the ability to modify out-of-game settings

k. DisplayGamesList

Actor:	Client
Preconditions:	The player is not in an active game but has chosen to join a game and is on the server list screen
Postconditions:	The player will see a list of games where he/she can select one to play
Flow:	The client will be updated to reflect the games list

l. DisplayPregameScreen

Actor:	Client
Preconditions:	The player has decided to join a game or has created a game
Postconditions:	The pre-game screen will be displayed to the player
Flow:	The player will see which characters will be joining him/her in the game

m. CreateGame

Actor:	Client
Preconditions:	The player is not in an active game
Postconditions:	The create game screen will be displayed
Flow:	The host player will have the ability to modify the game settings

n. JoinGame

Actor:	Client
Preconditions:	The player is not in an active game
Postconditions:	The join game screen will be displayed
Flow:	The player will have the ability to see which games are currently active

o. PlayGame

Actor:	Client
Preconditions:	There are at least three participants and the game is currently not active
Postconditions:	The player will be taken to the game screen
Flow:	The player will have the ability to play the game with other players

p. QuitGame

Actor:	Client
Preconditions:	There is an active game with participants
Postconditions:	The player will be taken to the main screen
Flow:	The player has exited the game

5. NotificationManager

The NotificationManager class is required to read in a language dictionary updated the player's settings.

a. ReadLanguageFiles

Actor:	System
Preconditions:	There is a directory on the sever that contains language files
Postconditions:	The notifications in the files have been read
Flow:	The system updates

b. GetSupportedLanguages

Actor:	System
Preconditions:	The language files have been read and processed
Postconditions:	Return the list of languages
Flow:	The system updates

c. GetRawNotification

Actor:	System
Preconditions:	The languages files have been read and processed and the user has selected a language
Postconditions:	The system now has the raw notification in the user's chosen language
Flow:	The system updates

PROJECT INTERFACE

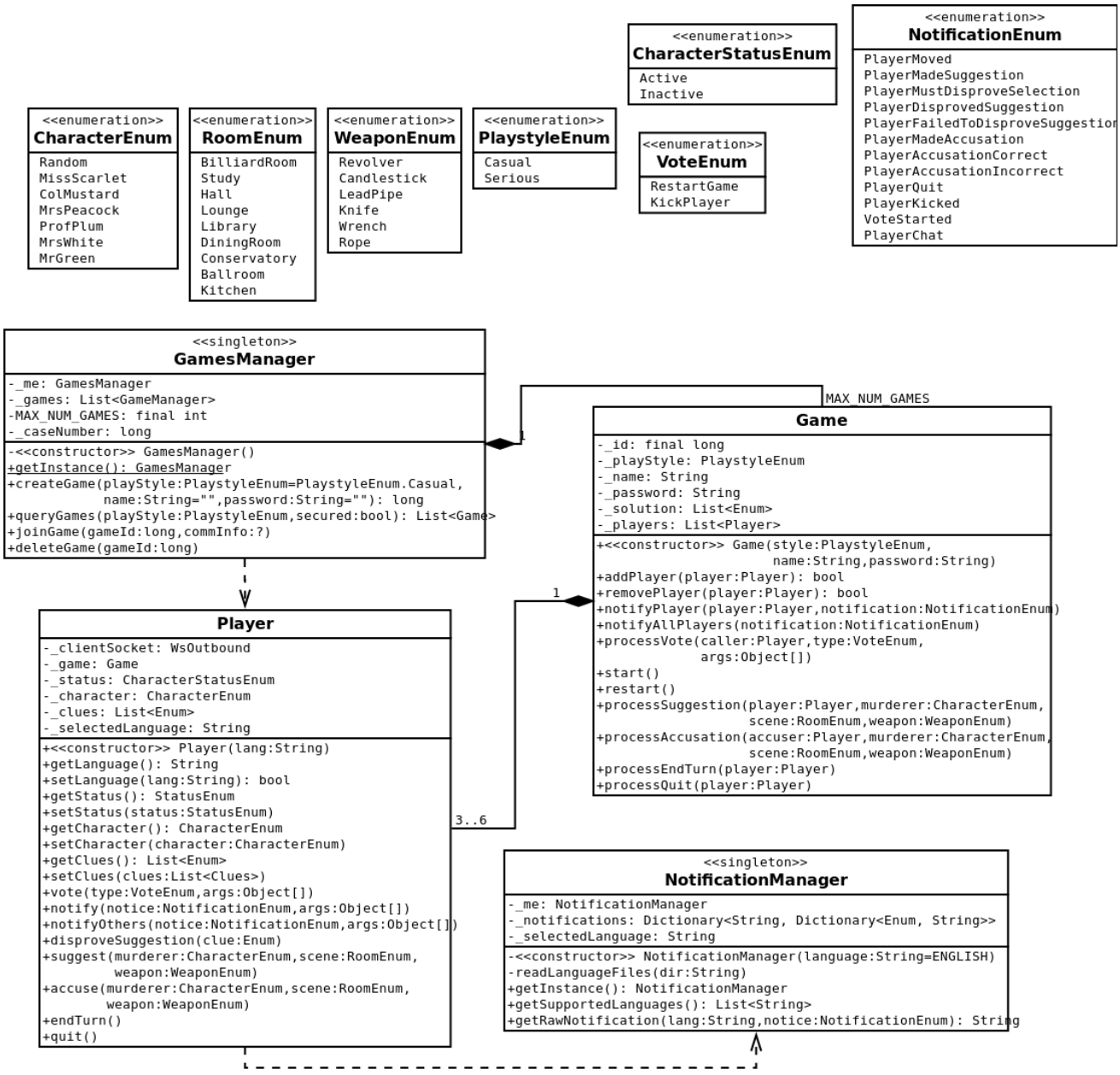
Clue-less can be broken down into four main interfaces:

1. User Interface: defined under "Project Scope" (pg. 3)
2. Hardware Interface: All devices with supported software interfaces
3. Software Interface: Modern Browsers (e.g., Mozilla Firefox v.5+, Google Chrome build 27+, and Internet Explorer v.10+) with JavaScript enabled.
4. Communications Interface: HTTP and WebSocket communication

To explain how these interfaces communicate with one another, a general flow of a message traveling through the system is described below.

1. The user performs an action in their browser.
2. The code-behind of the Client generates a JSON string
3. The string is sent over this browser's WebSocket to the Clueless server
4. The Clueless server processes the string
5. Based on the contents of the string, the server invokes a function in either the GamesManager, Game, Player, or NotificationManager.
6. A response JSON string is generated
7. The string is sent to either the invoking Client, another Client, or all Clients connected to the invoking browser's game.
8. The code-behind of each browser receiving the string processes it and takes appropriate action

UML DIAGRAM


<<enumeration>>
CharacterEnum
Random
MissScarlet
ColMustard
MrsPeacock
ProfPlum
MrsWhite
MrGreen
<<enumeration>>
RoomEnum
BilliardRoom
Study
Hall
Lounge
Library
DiningRoom
Conservatory
Ballroom
Kitchen
<<enumeration>>
WeaponEnum
Revolver
Candlestick
LeadPipe
Knife
Wrench
Rope
<<enumeration>>
PlaystyleEnum
Casual
Serious
<<enumeration>>
CharacterStatusEnum
Active
Inactive
<<enumeration>>
VoteEnum
RestartGame
KickPlayer
<<enumeration>>
NotificationEnum
PlayerMoved
PlayerMadeSuggestion
PlayerMustDisproveSelection
PlayerDisprovedSuggestion
PlayerFailedToDisproveSuggestion
PlayerMadeAccusation
PlayerAccusationCorrect
PlayerAccusationIncorrect
PlayerQuit
PlayerKicked
VoteStarted
PlayerChat
<<singleton>>
GamesManager
-_me: GamesManager
-_games: List<GameManager>
-MAX_NUM_GAMES: final int
-_caseNumber: long
-<<constructor>> GamesManager()
+getInstance(): GamesManager
+createGame(playStyle:PlaystyleEnum=PlaystyleEnum.Casual,
name:String="",password:String=""): long
+queryGames(playStyle:PlaystyleEnum,secured:bool): List<Game>
+joinGame(gameId:long,commInfo:?)
+deleteGame(gameId:long)
MAX_NUM_GAMES
1
Game
-_id: final long
-_playStyle: PlaystyleEnum
-_name: String
-_password: String
-_solution: List<Enum>
-_players: List<Player>
+<<constructor>> Game(style:PlaystyleEnum,
name:String,password:String)
+addPlayer(player:Player): bool
+removePlayer(player:Player): bool
+notifyPlayer(player:Player,notification:NotificationEnum)
+notifyAllPlayers(notification:NotificationEnum)
+processVote(caller:Player,type:VoteEnum,
args:Object[])
+start()
+restart()
+processSuggestion(player:Player,murderer:CharacterEnum,
scene:RoomEnum,weapon:WeaponEnum)
+processAccusation(accuser:Player,murderer:CharacterEnum,
scene:RoomEnum,weapon:WeaponEnum)
+processEndTurn(player:Player)
+processQuit(player:Player)
1
3..6
Player
-_clientSocket: WsOutbound
-_game: Game
-_status: CharacterStatusEnum
-_character: CharacterEnum
-_clues: List<Enum>
-_selectedLanguage: String
+<<constructor>> Player(lang:String)
+getLanguage(): String
+setLanguage(lang:String): bool
+getStatus(): StatusEnum
+setStatus(status:StatusEnum)
+getCharacter(): CharacterEnum
+setCharacter(character:CharacterEnum)
+getClues(): List<Enum>
+setClues(clues:List<Clues>)
+vote(type:VoteEnum,args:Object[])
+notify(notice:NotificationEnum,args:Object[])
+notifyOthers(notice:NotificationEnum,args:Object[])
+disproveSuggestion(clue:Enum)
+suggest(murderer:CharacterEnum,scene:RoomEnum,
weapon:WeaponEnum)
+accuse(murderer:CharacterEnum,scene:RoomEnum,
weapon:WeaponEnum)
+endTurn()
+quit()
<<singleton>>
NotificationManager
-_me: NotificationManager
-_notifications: Dictionary<String, Dictionary<Enum, String>>
-_selectedLanguage: String
-<<constructor>> NotificationManager(language:String=ENGLISH)
-readLanguageFiles(dir:String)
+getInstance(): NotificationManager
+getSupportedLanguages(): List<String>
+getRawNotification(lang:String,notice:NotificationEnum): String

DOCUMENT ACCEPTANCE

Member	Signature
Shan Sabri	_____
Hartanto Thio	_____
Davis Gigogne	_____