View Diagram Description

**MainMenuView**

**Description:** MainMenuView is a public class that is responsible for creating the graphical user interface buttons. These buttons will be used to allow the user either to create or join a game, or leave the service all together.

**Fields:**

* Button createGameButton: Used for creating a new game.
* Button joinGameButton: Used for joining an existing game.
* Button exitGameButton: Used for terminating the service.

**Methods:** As needed to handle the fields.

**CreateGameView**

**Description:** Public class responsible for creating graphical user interface window, where the host-player can select game settings prior to creating a game.

**Fields:**

* Button[4] numPlayers: List of Button instances representing the number of players allowed in the game. Minimum of two, and maximum of five.
* TextField timeOut: Stores the user input for the time out limit.
* TextField nsid: Represents the identification of the host-player.
* Button force: Allows the host-player to override an existing game under his/her identification.
* Button createButton: Allows the host-player to create a game with the selected settings.
* Button backButton: Gives the host-player the option to go back to the MainMenuView.

**Methods:** As needed to handle the fields.

**JoinGameView**

**Description:** Public class responsible for creating graphical user interface window, where the player can specify which game he/she would like to join.

**Fields:**

* TextField gameID: Used to specify which game the user wants to join.
* TextField token: Used to verify if the player is allowed to join the game.
* TextField nsid: Represents the identification of the player.
* Button joinButton: Allows the player to join a game specified by gameID and token.
* Button backButton: Gives the player the option to go back to the MainMenuView.

**Methods:** As needed to handle the fields.

**LobbyView**

**Description:** Public class responsible for creating graphical user interface window, where the players are grouped prior to starting the game.

**Fields:**

* String gameID: String value that represents the identification for the game.
* String token: String value that represents the needed verification to join the game.
* int numPlayers: Integer value that represents the number of players.
* Button addAIButton: Gives the host-player the option to add a computer-player to the game.
* Button leaveButton: Gives the player the option to go back to the MainMenuView.

**Methods:** As needed to handle the fields.

**GameView**

**Description:** Public class responsible for creating graphical user interface window, which shows all players, cards, and available options.

**Fields:**

* int numPlayers: Keeps track of all active players in the game.
* Button[4..5, 2..5] cardButtons: Two dimensional list that represents the number of cards that each player has.
* Button toggleDiscardButton: Allows the player to discard a card.
* Button toggleLogButton: Allows the player to activate/de-activate the Log.
* Button leaveButton: Gives the player the option to go back to the MainMenuView.

**Methods:** As needed to handle the fields.

**GameOverView**

**Description:** Public class responsible for creating graphical user interface window, which displays the final score achieved by the players and a message to describe that achievement.

**Fields:**

* int score: Integer value that represents the score achieved by the players.
* Button mainMenuButton: Gives the player the option to go back to the MainMenuView.
* Button quitButton: Used for terminating the service.

**Methods:** As needed to handle the fields.

**Panel**

**Description:** Public class responsible for creating panels for each graphical user interface view. These panels will be used to activate/de-activate a view instead of re-creating the display each time.