CSCI 4448 Project Part 2
Carlos Lawrence, Parth Mishra, David Brunelle

Team: Parth Mishra

Title: PlayGo!

Project Summary: An implementation of the ancient game of “Go”. This project is inspired by Mark Zuckerberg’s team at Facebook creating an AI that defeated the top level players. For this implementation, it will allow two human players to play against each other. Time permitting, an elementary AI that can make legal moves would be a reach goal.

Project Requirements:
*****No business requirements are needed

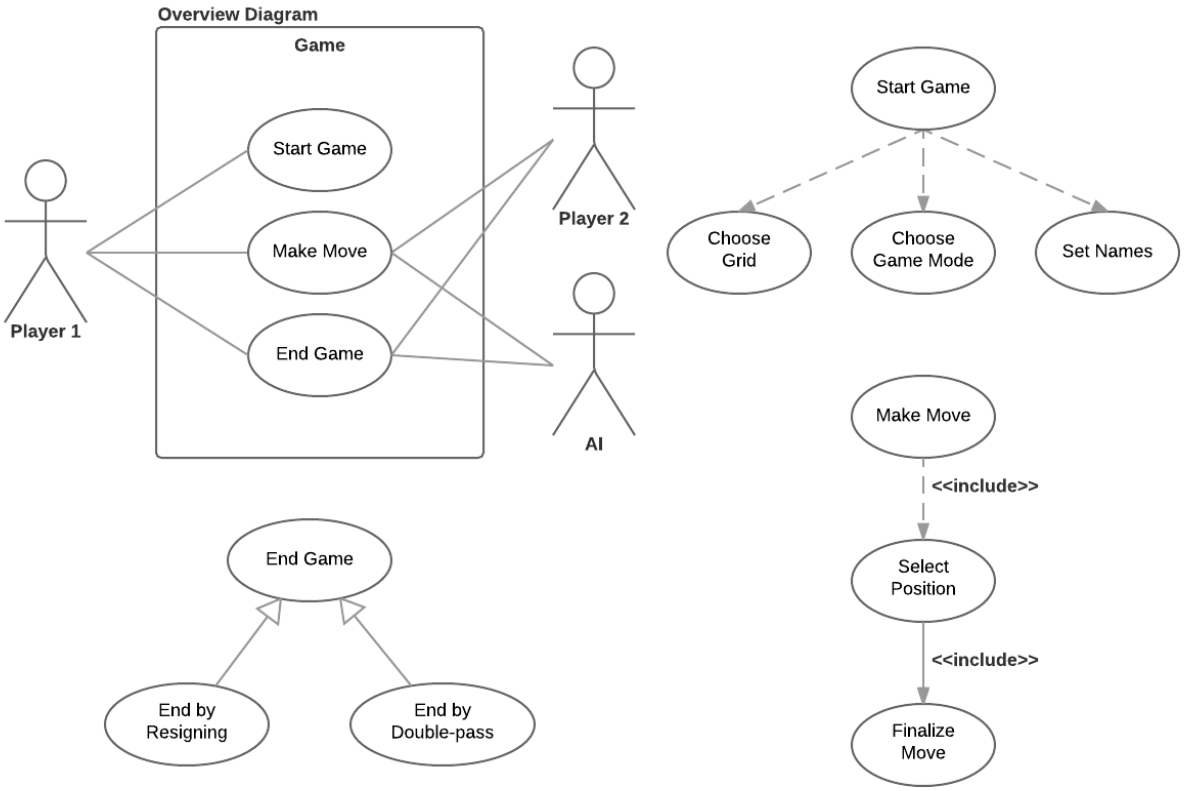
User Requirements			
ID	**Requirements**	**User**	**Priority**
UR-001 Carlos	Players can start a game on a 9x9, 11x11, or 19x19 board	Player	Med
UR-002 Parth	Players can pass if they don’t want to make a move	Player	Med
UR-003 Parth	Players can resign at any point	Player	Med
UR-004 Dave	Players can click a corner on the grid to show where their piece would be added to the board	Player	High
UR-005 Carlos	Players can set their name at the start of the game	Player	Low
UR-006 Dave	Players click submit to finalize each move	Player	High

Functional Requirements			
ID	**Requirements**	**User**	**Priority**
FR-001	Players may only make legal moves	Game	High

FR-002	Captures pieces are removed from the game	Game	High
FR-003	Player's territory and number of captured pieces are tracked	Game	High
FR-004	The game ends when both players pass consecutively	Game	Critical

Non-functional Requirements			
ID	**Requirements**	**User**	**Priority**
NFR-001	An AI is available for players to play against	AI	Low
NFR-002	The screen displays the results of the previous 5 games	UI	Low

Use Cases:



Use Case ID:	UC-001		
Use Case Name:	Choose Board		
Description:	Players can start a game on a 9x9, 11x11, or 19x19 board		
Actors:	Player 1		
Pre-conditions:	Player 1 has elected to start a game		
Post-conditions:	The game board size is set and the game board initialized		
Frequency of Use:	Every time a new game is started, only once per game		
Flow of events:		**Actor Action**	**System Response**
	1	Starts Game	Provide list of game options to select from
	2	Select 9x9 Size	Generate a 10x10 table to represent the game board
	3	Begin	Display board using selected options
Variations:		**Actor Action**	**System Response**
	2	Select 11x11 Size	Generate a 12x12 table to represent the game board
	2	Select 19x19 Size	Generate a 20x20 table to represent the game board
Notes and Issues:	Player must select one option from each category before beginning the game		
Developer Notes:			

Use Case ID:	UC-002
Use Case Name:	Passing

Description:	Players can choose to pass if they don't want to make a move on their turn		
Actors:	Player 1, Player 2, System		
Pre-conditions:	The game has started and at least one move has been made		
Post-conditions:	The game end if the 2nd player chooses to pass, or play resumes with original player		
Frequency of Use:	One of two ways to possibly end the game		
Flow of events:		**Actor Action**	**System Response**
	1	Player 1 chooses to Pass	
	2	Player 2 chooses to Pass	End Game and display final score
Variations:			
	2	Player 2 may choose to make another move instead of passing	
Notes and Issues:	After a player has passed, the second player may not resign, they can only move or pass if they wish to end the game		
Developer Notes:			

Use Case ID:	UC-003
Use Case Name:	Resigning
Description:	Players can resign at any time after the first move and the preceding move was not a pass
Actors:	Player 1, Player 2
Pre-conditions:	The game has started and the initial move has been made and the previous move was not a pass
Post-conditions:	The game ends if either player chooses to resign

Frequency of Use:	One of two ways to end the game		
Flow of events:		**Actor Action**	**System Response**
	1	Player 1 resigns	End game and display score
Variations:		**Actor Action**	**System Response**
	1	Player 2 resigns	
Notes and Issues:	Players may only resign during their turn		
Developer Notes:			

Use Case ID:	UC-004		
Use Case Name:	Select Position		
Description:	Players left click a position on the board to lay down a stone		
Actors:	Player 1, Player 2, AI		
Pre-conditions:	The game has been initialized and there are valid positions on the board		
Post-conditions:	The board is updated showing the new stone		
Frequency of Use:	Whenever a user clicks a position on the board		
Flow of events:		**Actor Action**	**System Response**
	1	Select position	Verifies that position is valid
Variations:		**Actor Action**	**System Response**
	1	Select invalid position	Returns warning to user
	1	Select valid position	Redraw the board with the new stone in place
	1	AI selects position	Position assigned according to valid positions only

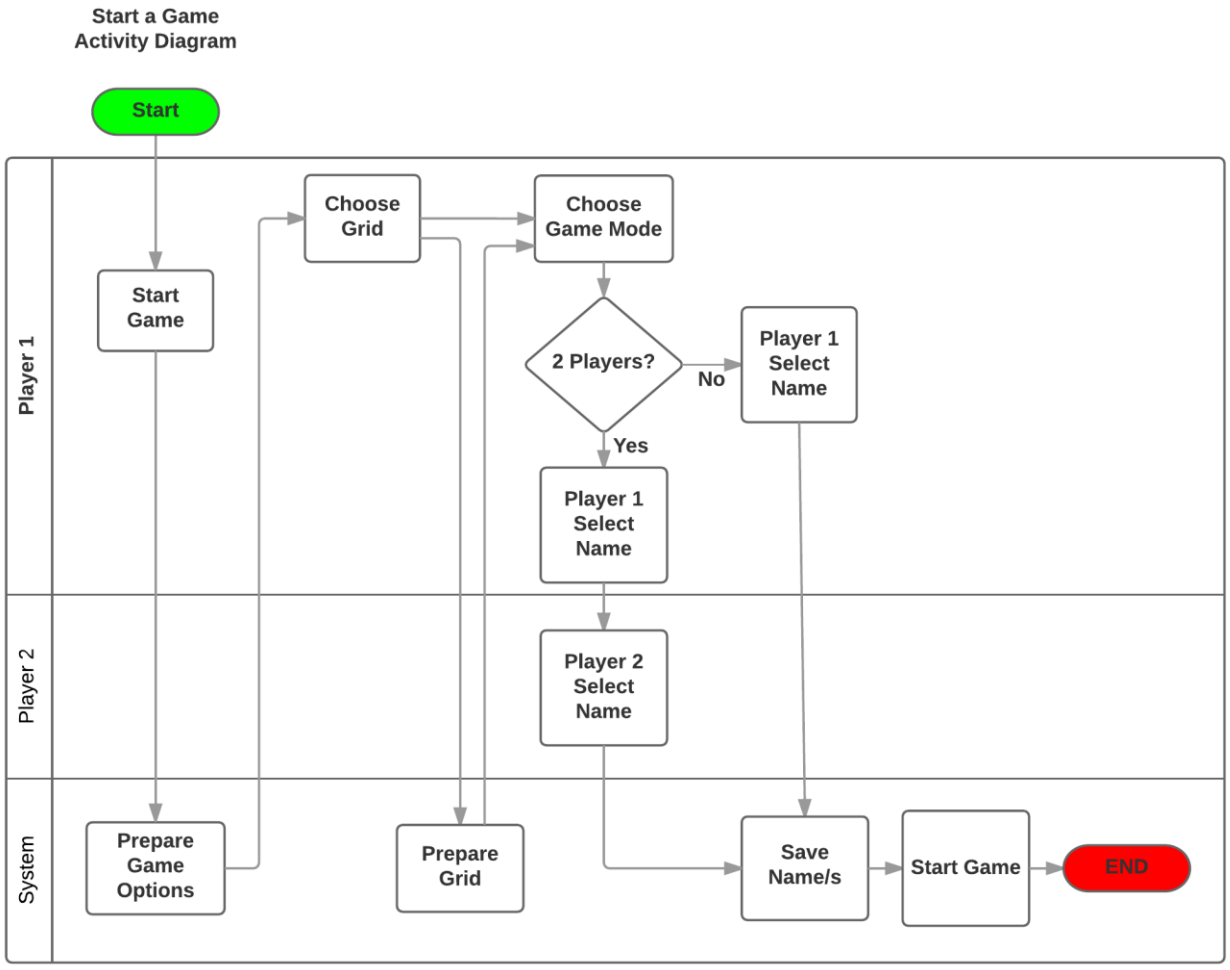
Notes and Issues:	
Developer Notes:	No graphics should be invoked until position is deemed to be valid.

Use Case ID:	UC-005		
Use Case Name:	Choose Name		
Description:	Players can choose their display name for recorded scores		
Actors:	Player 1, Player 2		
Pre-conditions:	Player 1 has elected to start a game and decided		
Post-conditions:	Each player has a unique name to distinguish their score		
Frequency of Use:	Every time a new game is started, only once per game		
Flow of events:		**Actor Action**	**System Response**
	1	Starts Game	Provide list of game options to select from
	2	Select 2 Player Game Mode	Present 2 labeled boxes for each player's name
	3	Begin	Display board using selected options
Variations:		**Actor Action**	**System Response**
	2	Select AI Game Mode	Present a labeled box for Player 1's name
Notes and Issues:	Names must be unique and appropriate		
Developer Notes:			

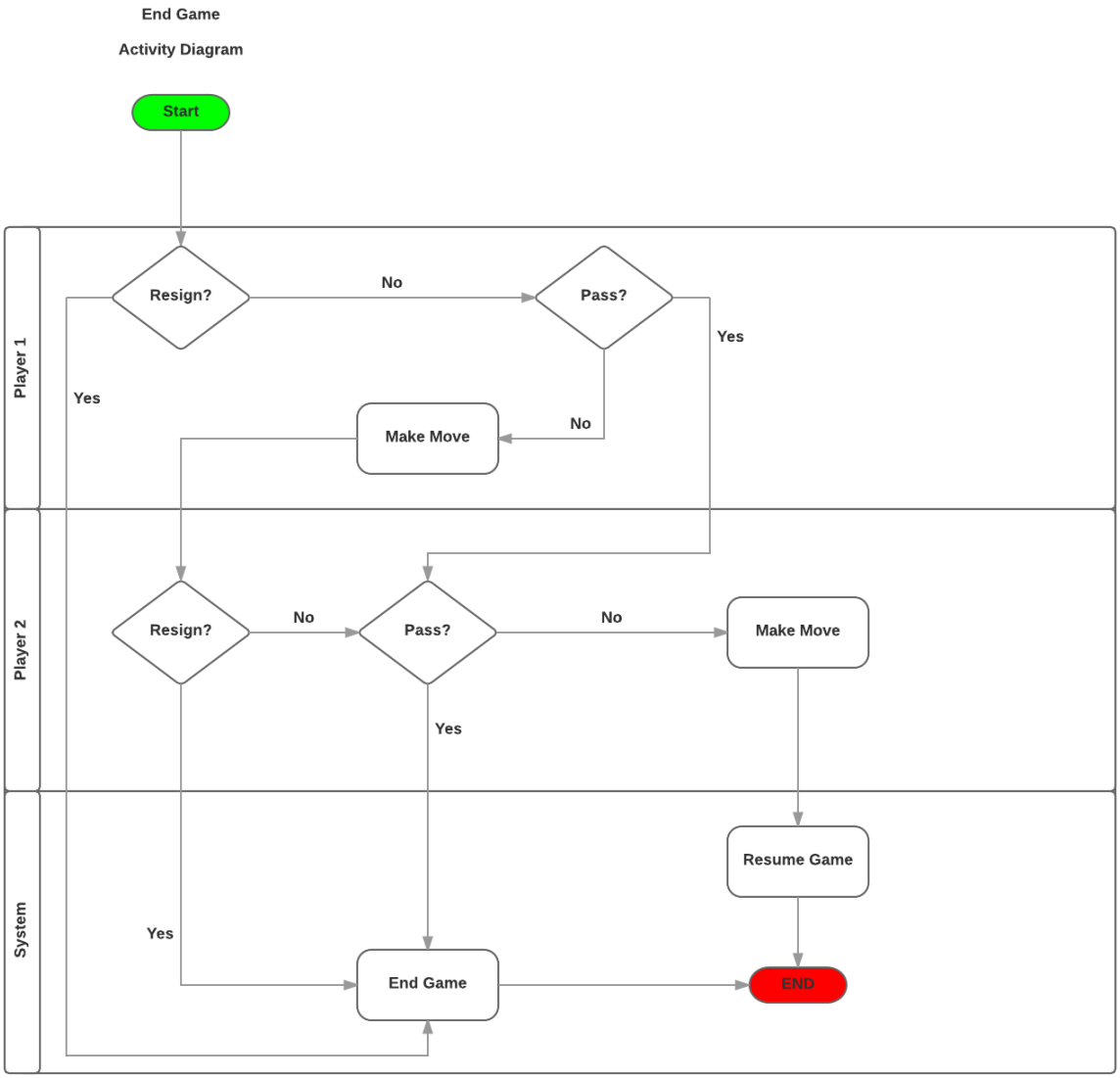
Use Case ID:	UC-006
Use Case Name:	Finalize move

Description:	Players click submit button to finish their turn		
Actors:	Player 1, Player 2, AI		
Pre-conditions:	The game has been initialized and a player has just selected a position on the board		
Post-conditions:	The state of the position is no longer valid for future moves		
Frequency of Use:	Any time a player makes a move		
Flow of events:		**Actor Action**	**System Response**
	1	Submit move	State of the selected position is changed from valid to invalid
Variations:		**Actor Action**	**System Response**
	1	AI submit	Move submitted as soon as valid position is calculated
Notes and Issues:			
Developer Notes:			

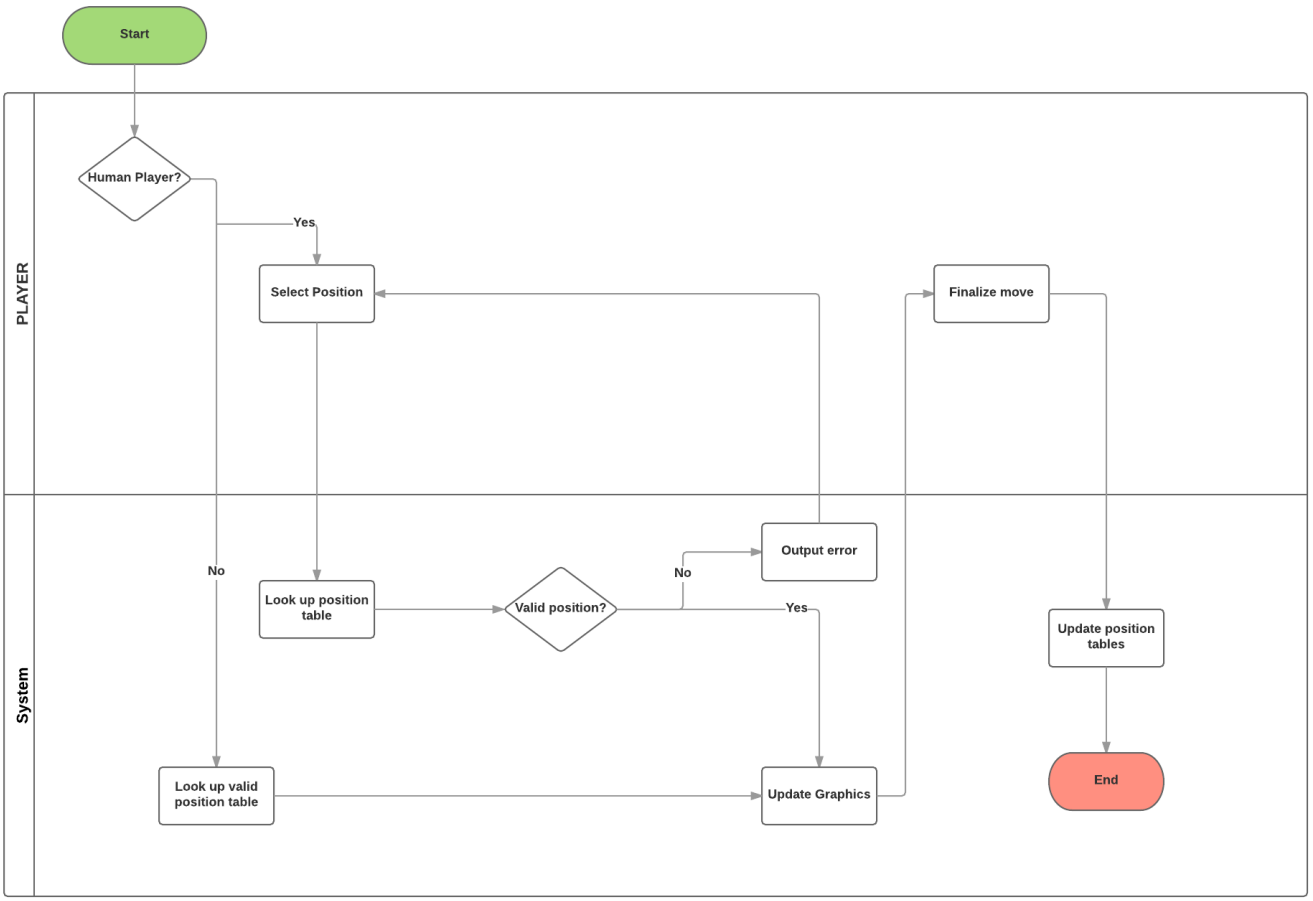
Activity Diagram:



Name: Carlos Lawrence
Use Case ID #: UC-002, UC-005
Use Case Name: Starting the Game

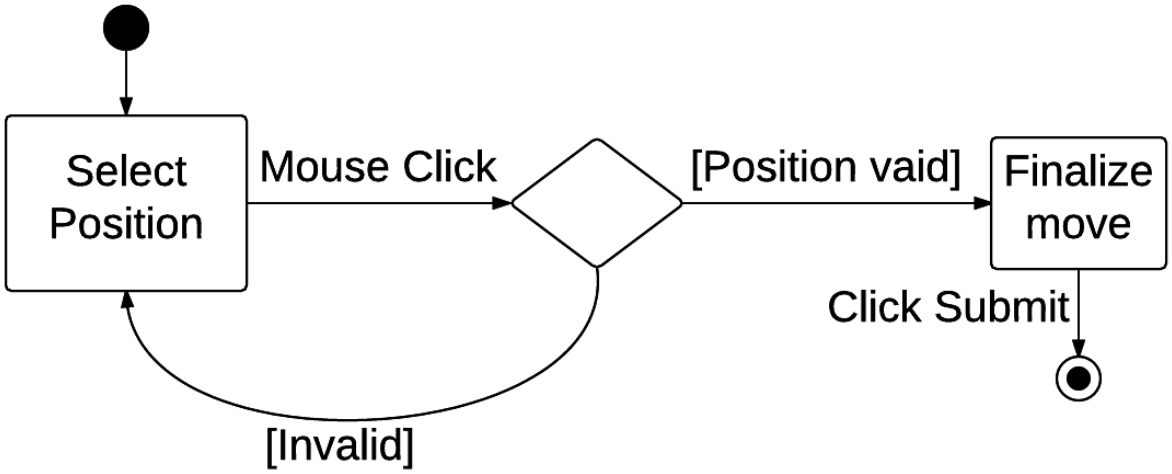


Name: Parth Mishra
Use Case ID #: UC-003, UC-004
Use Case Name: Possible Paths for Ending Game



Name: David Brunelle
Use case ID #: UC-004, UC-006
Use case Name: Make Move

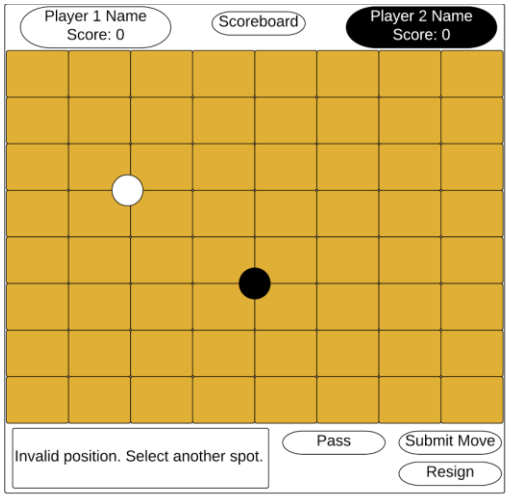
State Diagram:



Data Storage: Flat text file

Classes: ScoreBoard - Class in charge of reading from and writing to text file. If there is sufficient time, we may instead choose to store our data on a server to allow for a web based application.

UI Mockups:

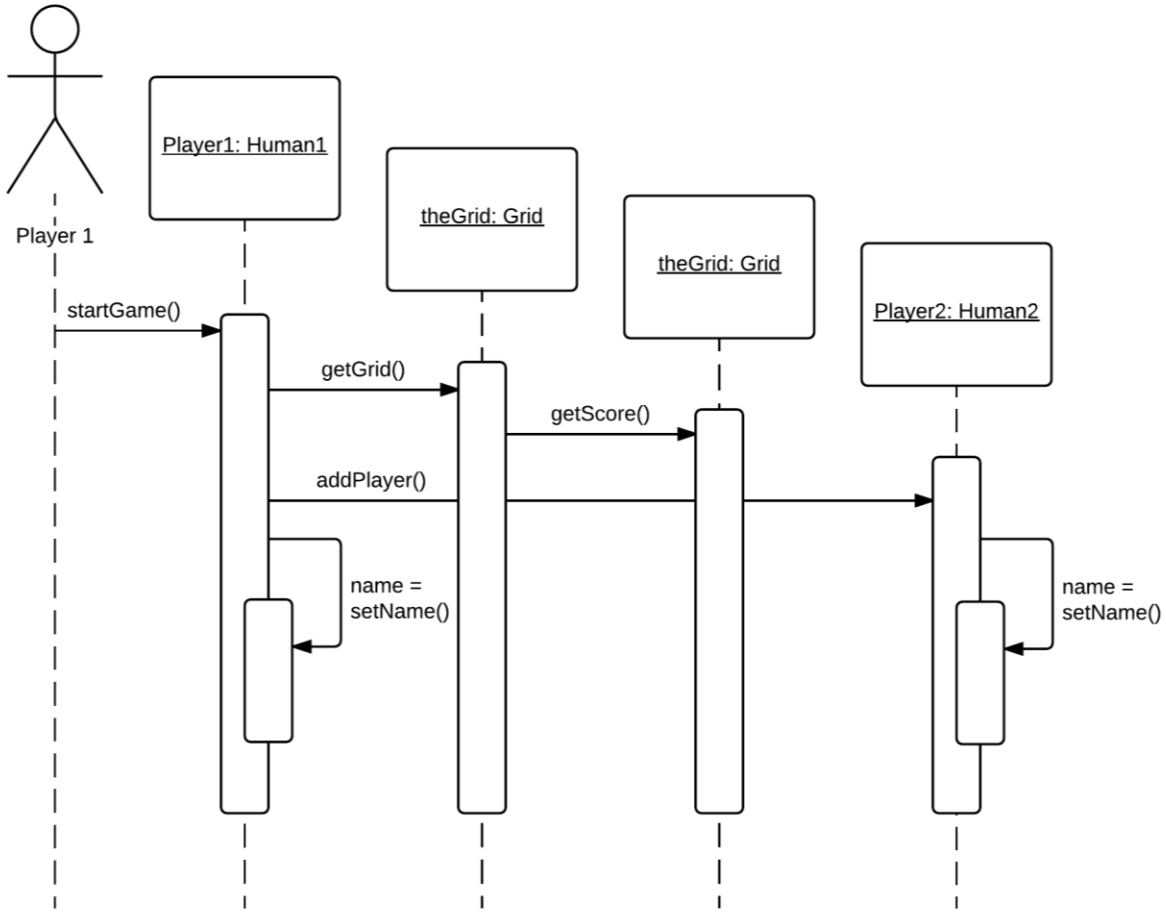


The first screen is an example of starting the game. Player 1 is given the options to start a new game with different board sizes and game modes, then each player has the ability to choose their name. Once all of the game options are selected, the player clicks the start button to move to the next screen

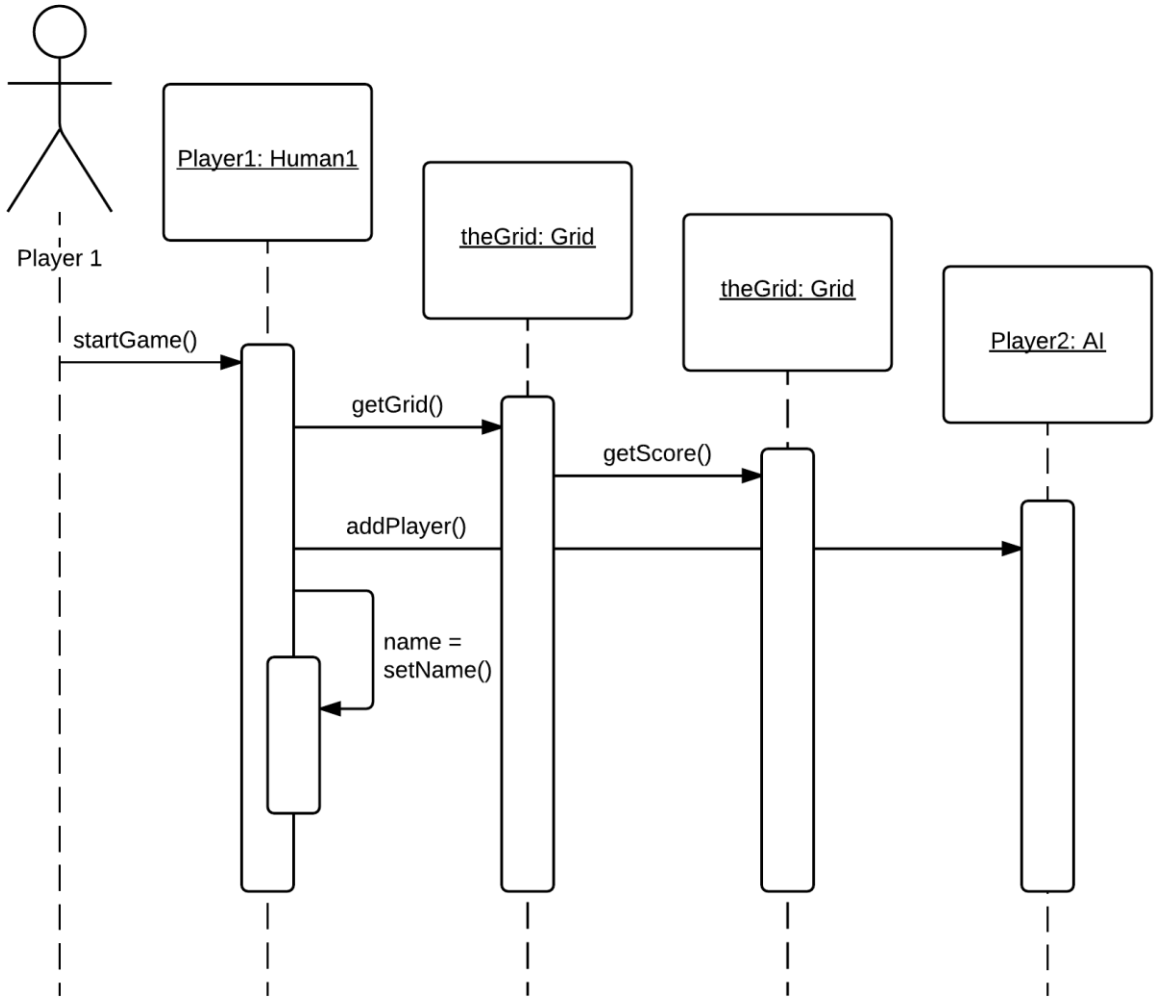
The next screen is an example of placing a piece on the board. Both player's names are displayed at the top along with an option to view the scoreboard. The majority of the screen is taken by the grid itself which displays all pieces that have been placed and a movable piece to be placed. The bottom left calculates the validity of each move and displays information on each move. Finally, the bottom right has options to finalize your move, pass, or resign.

The final screen is what gets displayed when the game ends. It contains options to start a new game or to quit.

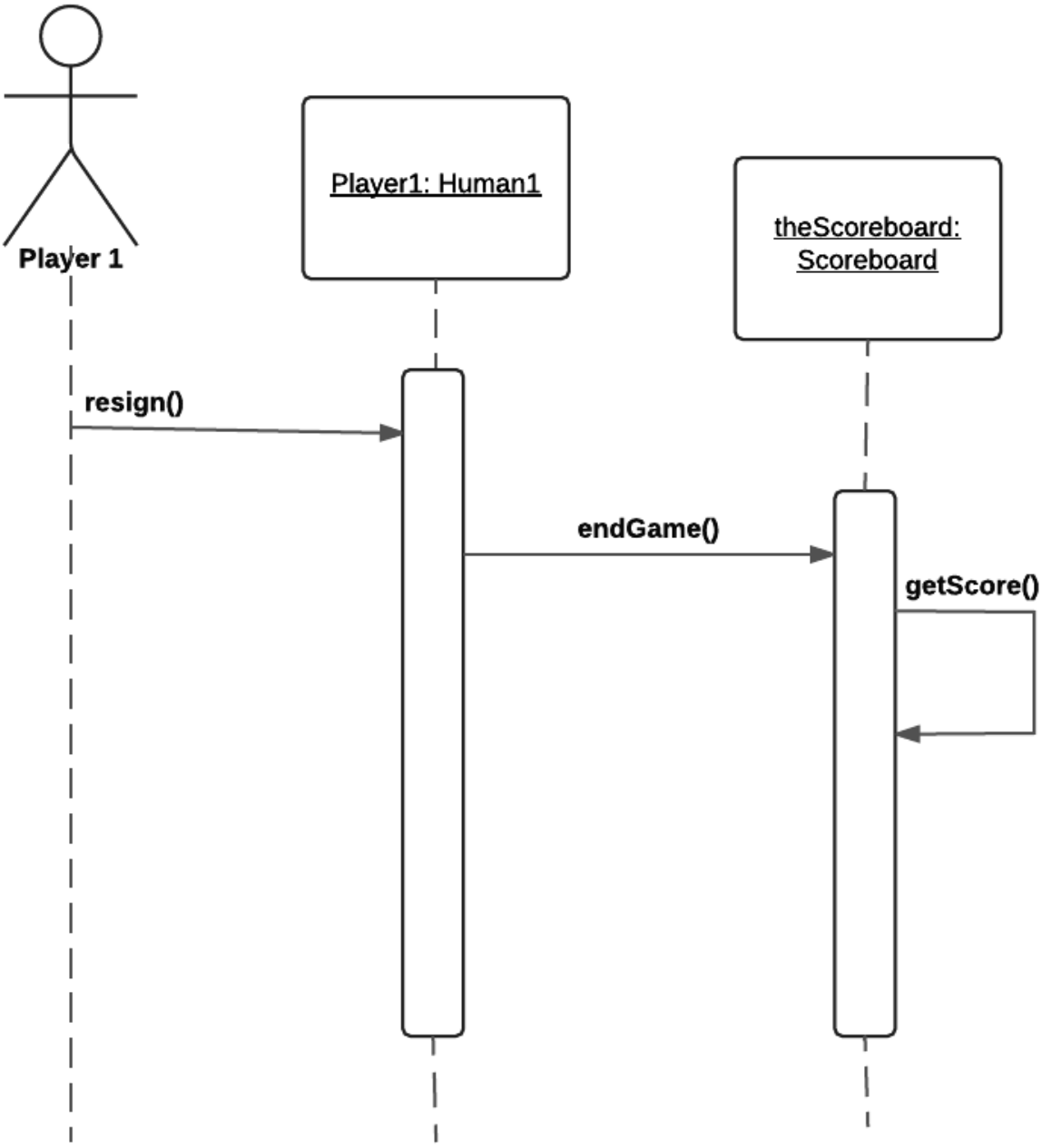
User Interactions:



Name: Carlos Lawrence
Use Case ID #: UC-001, UC-005
Requirement ID #: UR-001, UR-005
Use Case Name: Starting the Game with 2 Players
When the user starts the game, they create the object Human1, giving them the ability to start the game. As they select the game options, the different objects which the game uses are created for use. This follows a fork diagram where most of the dynamic behavior is placed in Human1, allowing them to be the control object.

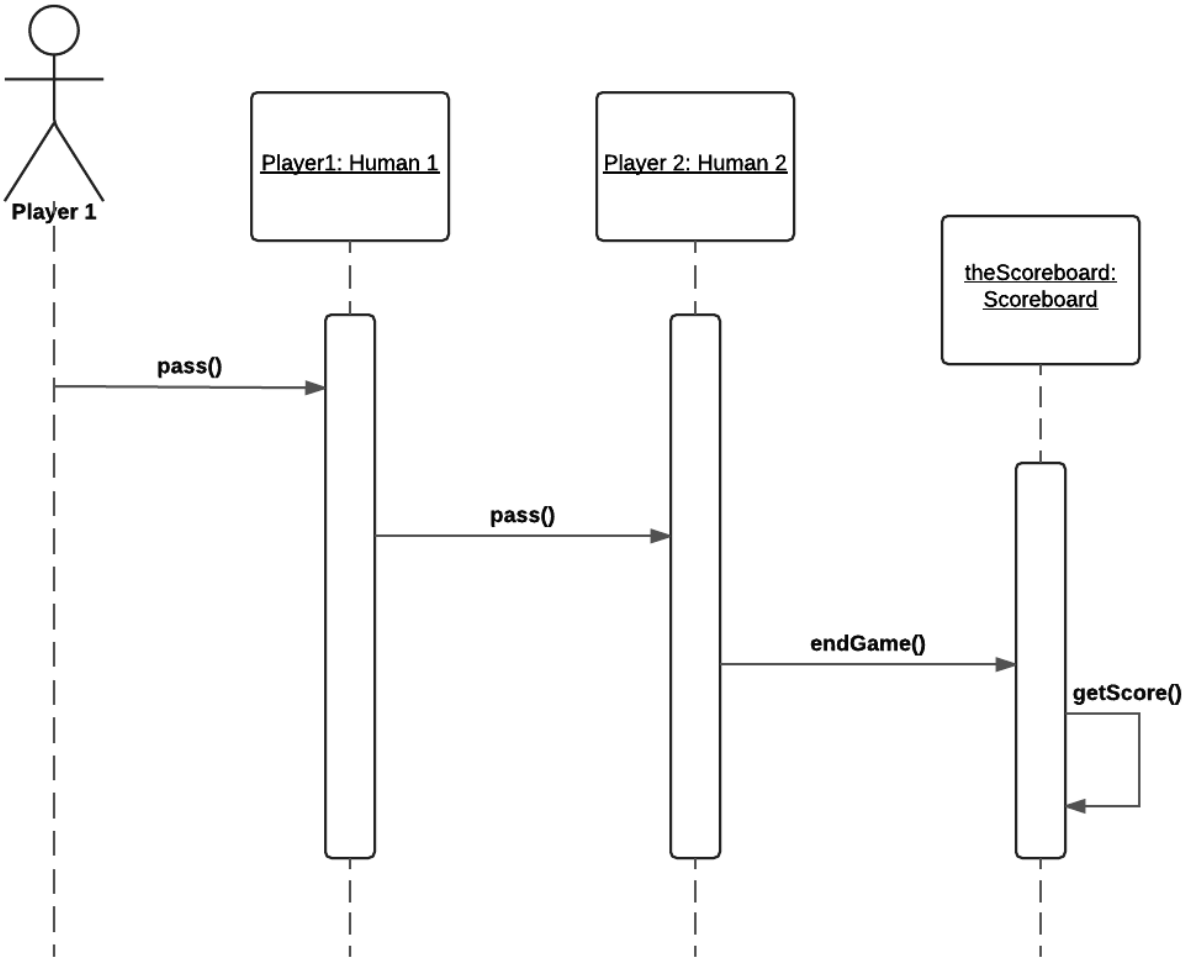


Name: Carlos Lawrence
Use Case ID #: UC-001, UC-005
Requirement ID #: UR-001, UR-005
Use Case Name: Starting the Game with AI
Very similar to starting the game with two players, except for the AI will always have the same name and does not need that to be set.



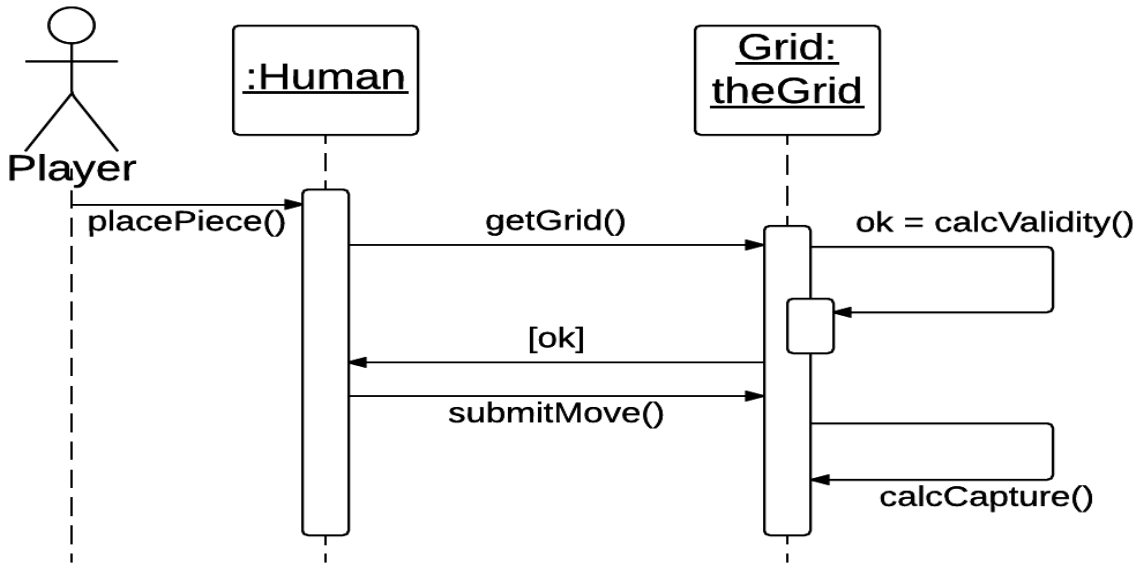
Name: Parth Mishra
Use Case ID #: UC-003, UC-004
Requirement ID #: UR-003
Use Case Name: Ending the Game via Resignation

A simple description of the scenario where a player elects to resign on their turn. When the player elects to resign, the resign() method calls upon the game state to execute the endGame() method thus ending the game and displaying the final score via getScore()

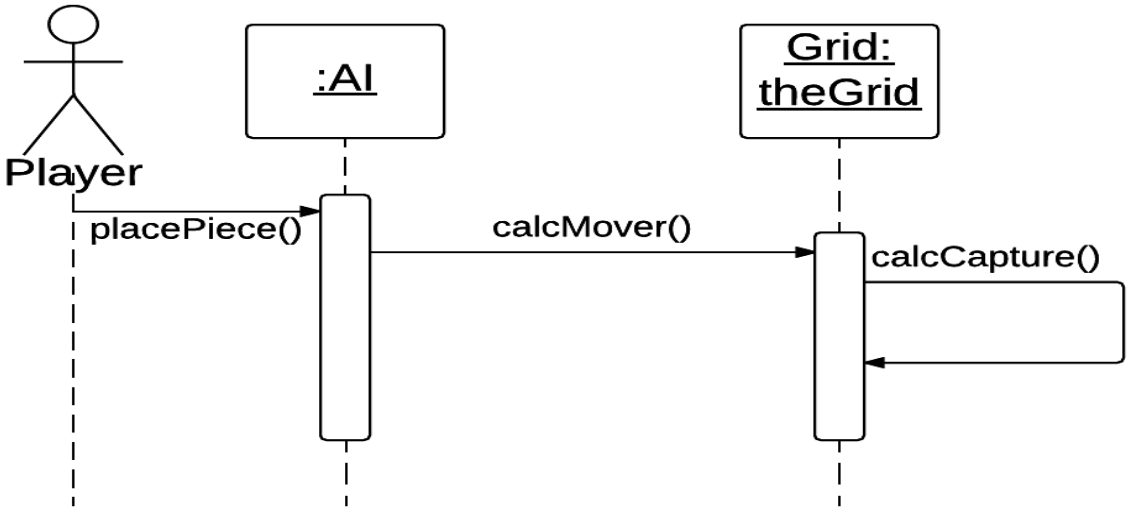


Name: Parth Mishra
Use Case ID #: UC-003, UC-004
Requirement ID #: UR-002
Use Case Name: Ending the Game via Consecutive Passes

The other potential method to ending a game which requires two consecutive passes. Note, after a player passes, the following player can't resign, only pass or move.



Name: David Brunelle
Use Case ID #: UC-004, UC-006
Requirement ID #: UR-004, UR-006, FR-001, FR-002
Use Case Name: Make Move (human)
In this case, a human player decides to play a piece within the confines of the board. This then calls upon the grid which assesses the validity of the move. Should the move be valid, control is returned to the human player so that it may be submitted. Upon submission, the grid calculates whether the move results in a capture or not.



Name: David Brunelle
Use Case ID #: UC-004, UC-006
Requirement ID #: NFR-001

Use Case Name: Make Move (AI)
In this alternative case, the AI player calculates a valid move before calling the grid. Following this, the grid determines whether or not this move results in a capture event.

Class Diagram: