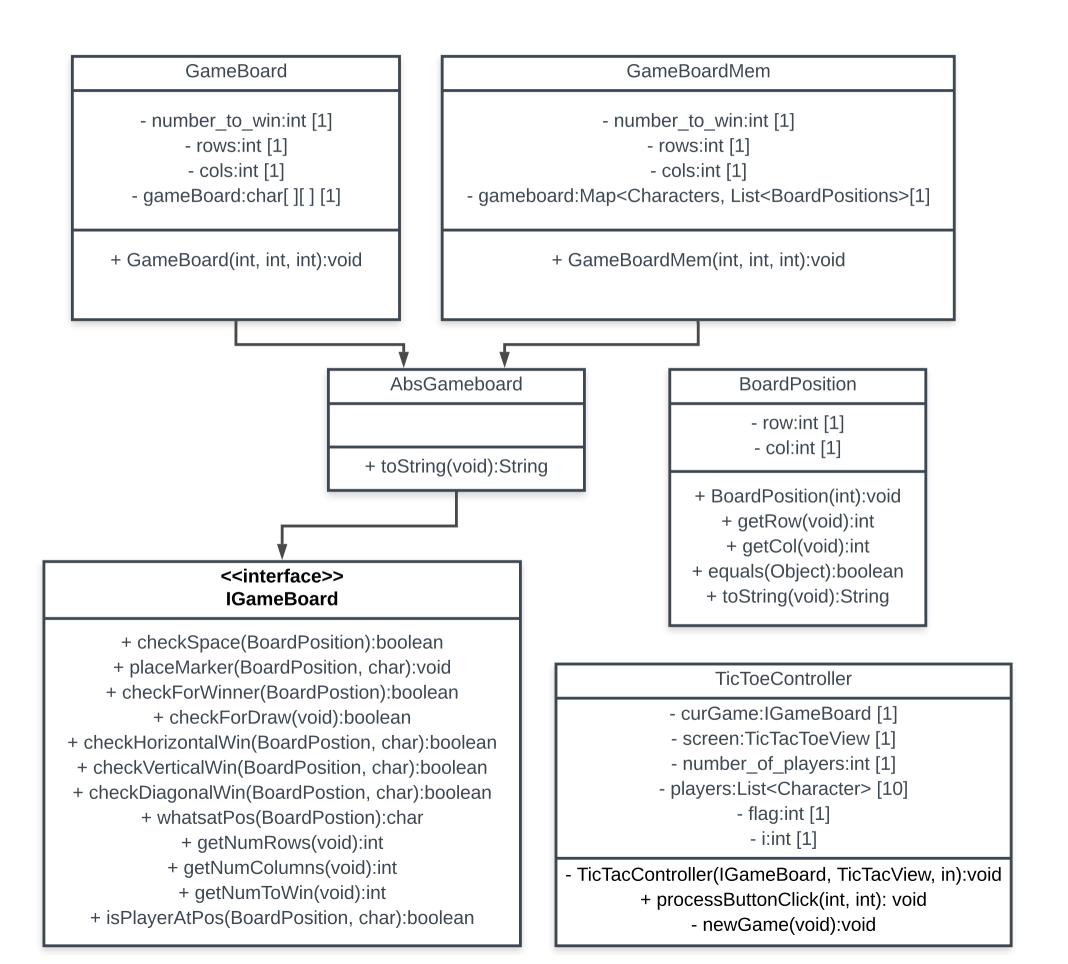
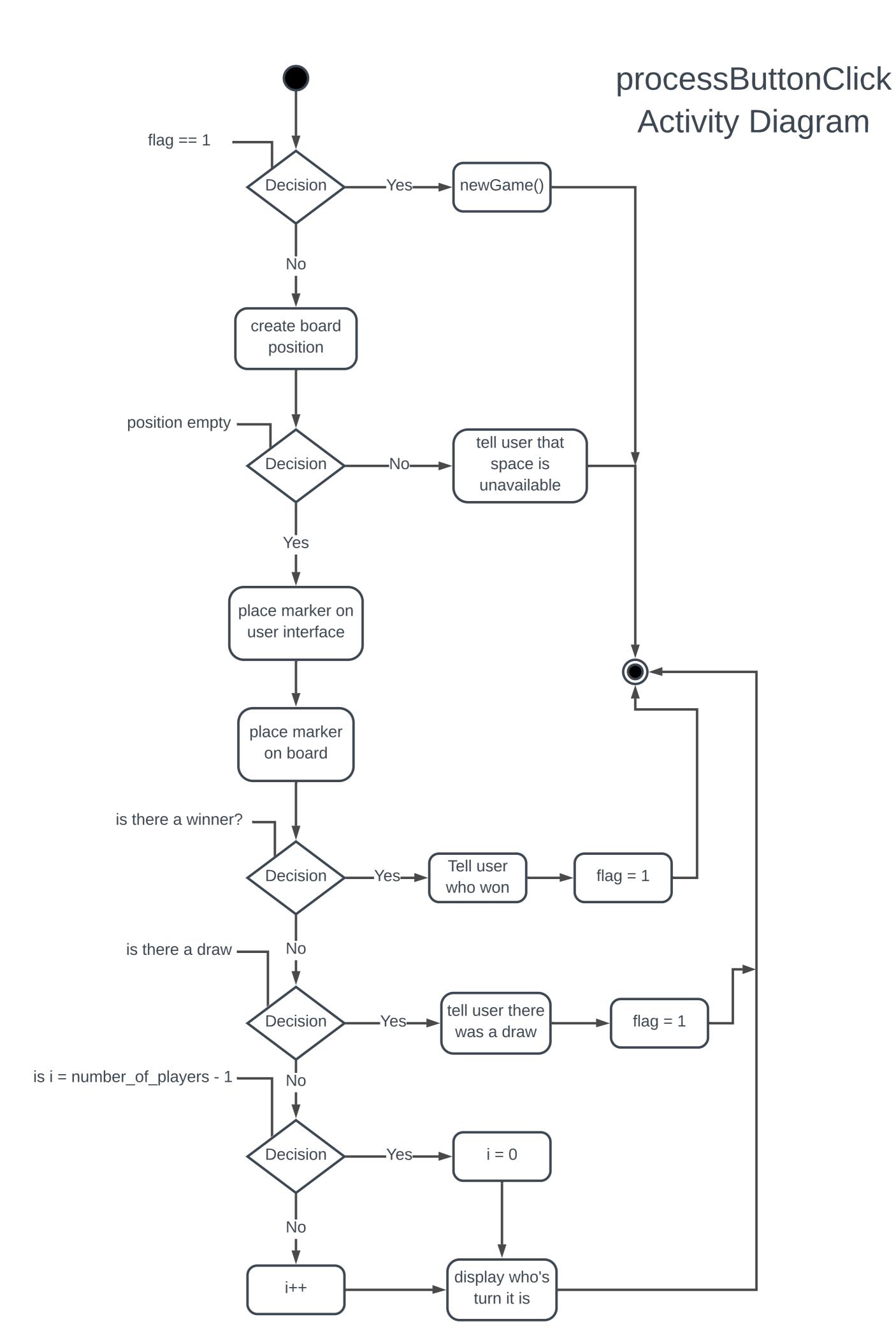
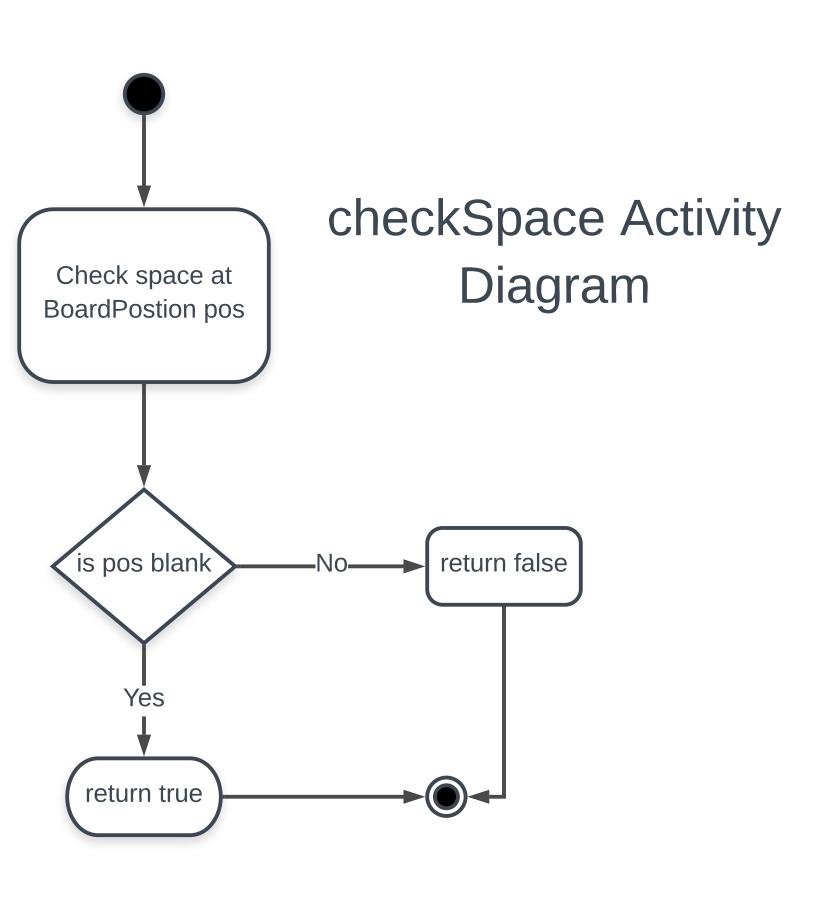
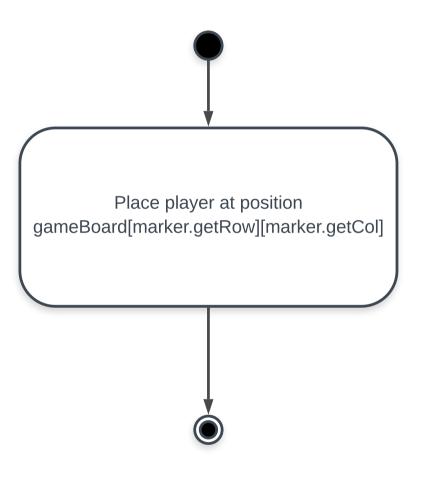
Class Diagrams

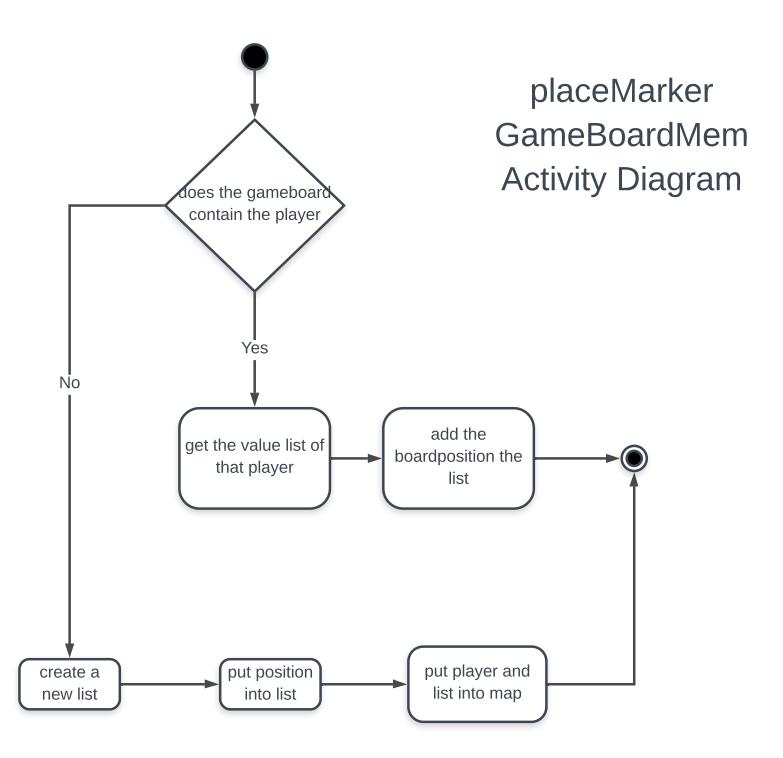


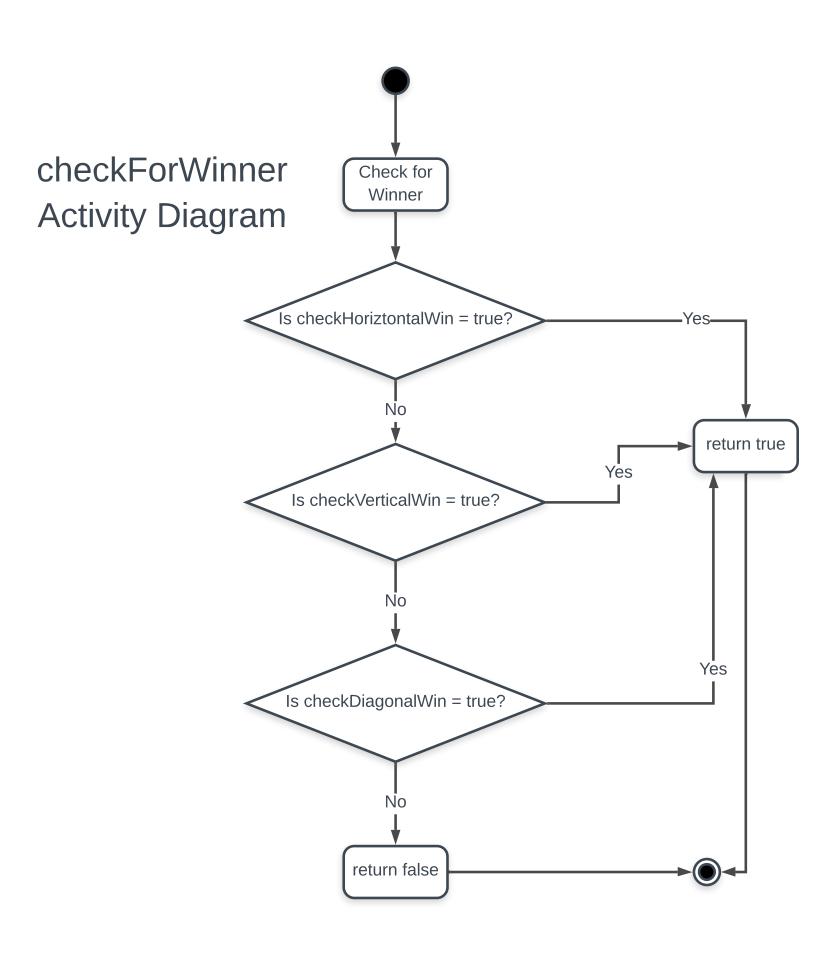




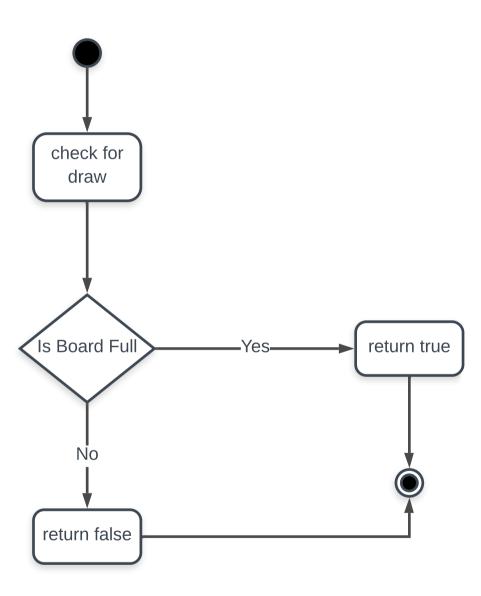


placeMarker Activity Diagram

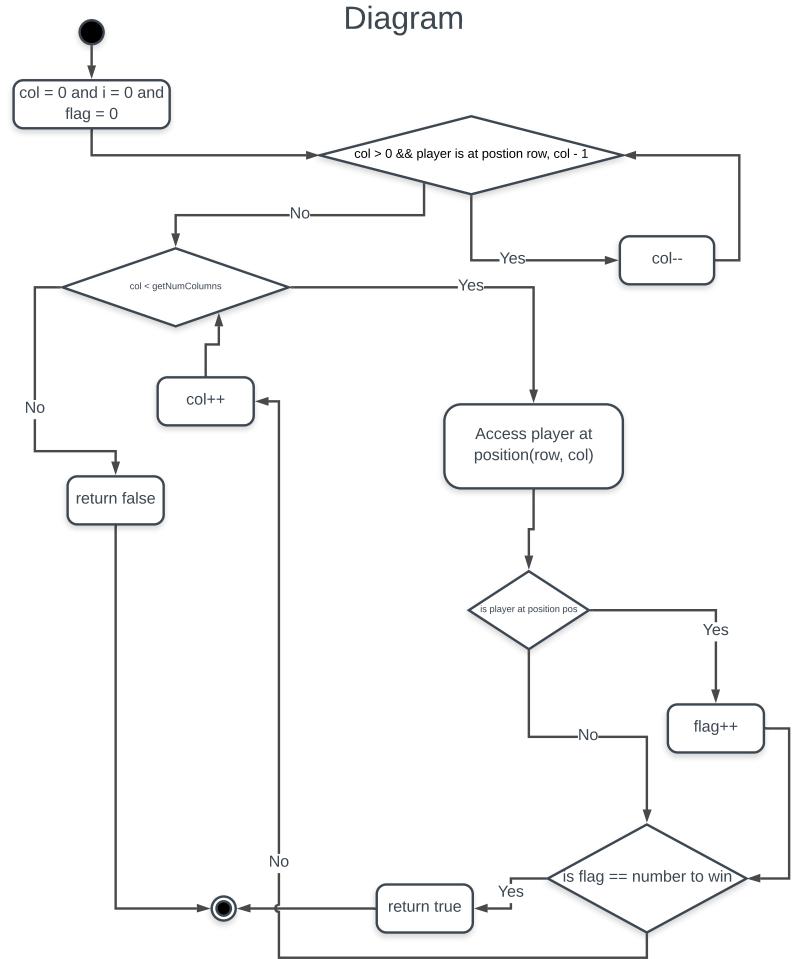




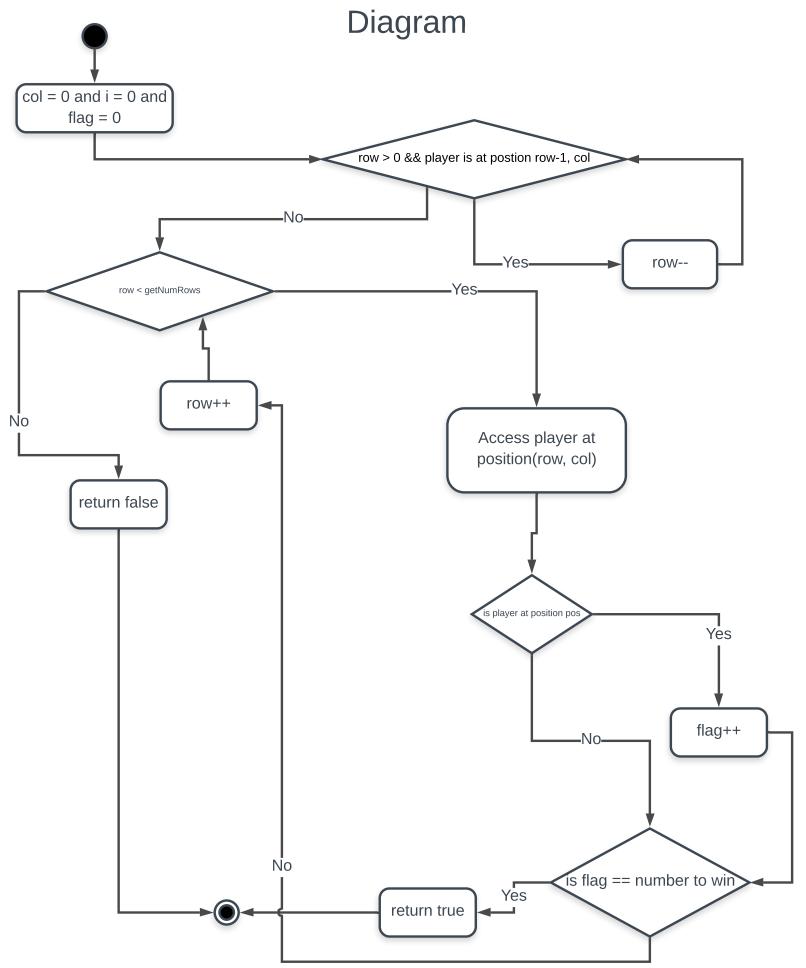
checkForDraw Activity Diagram



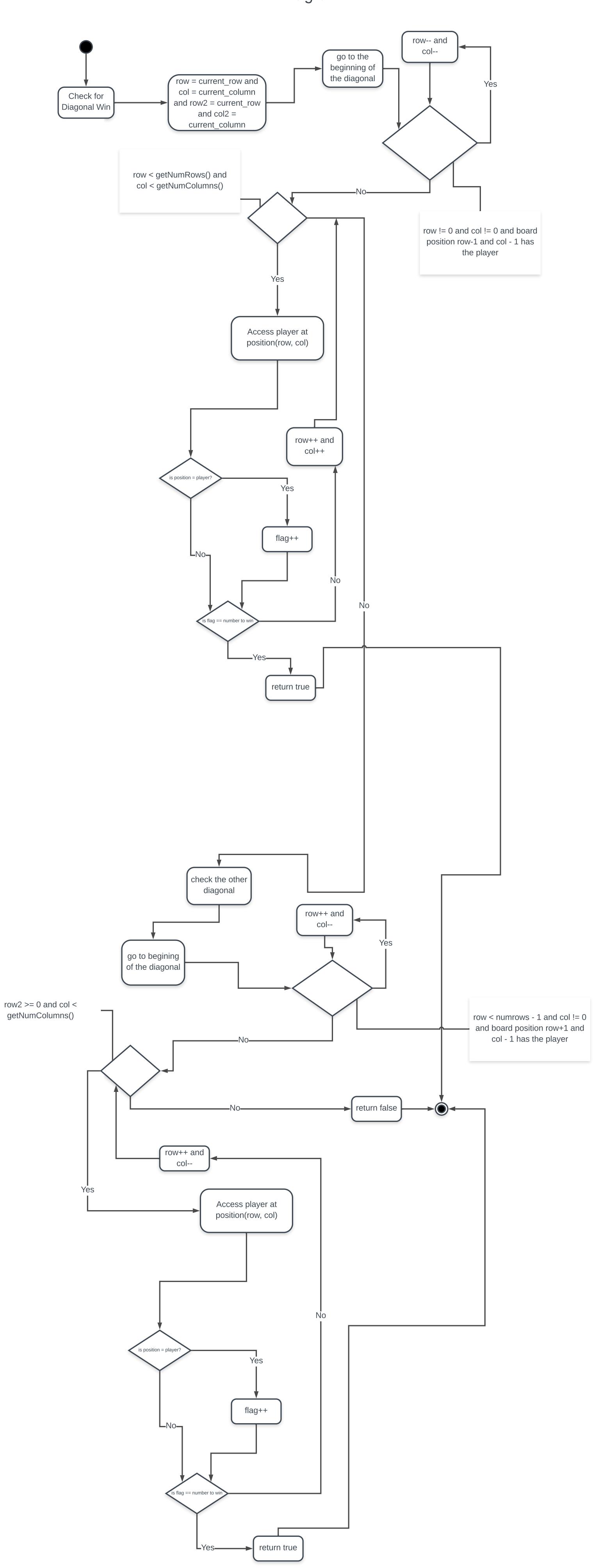
checkHorizontallWin Activity Diagram

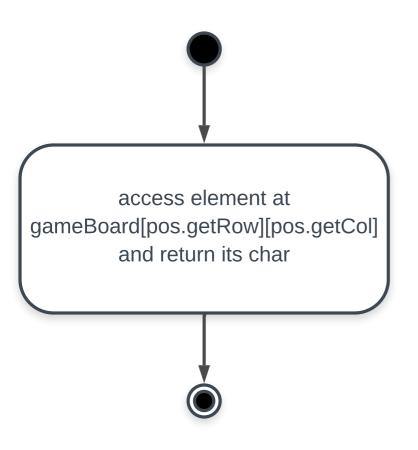


checkVerticalWin Activity Diagram

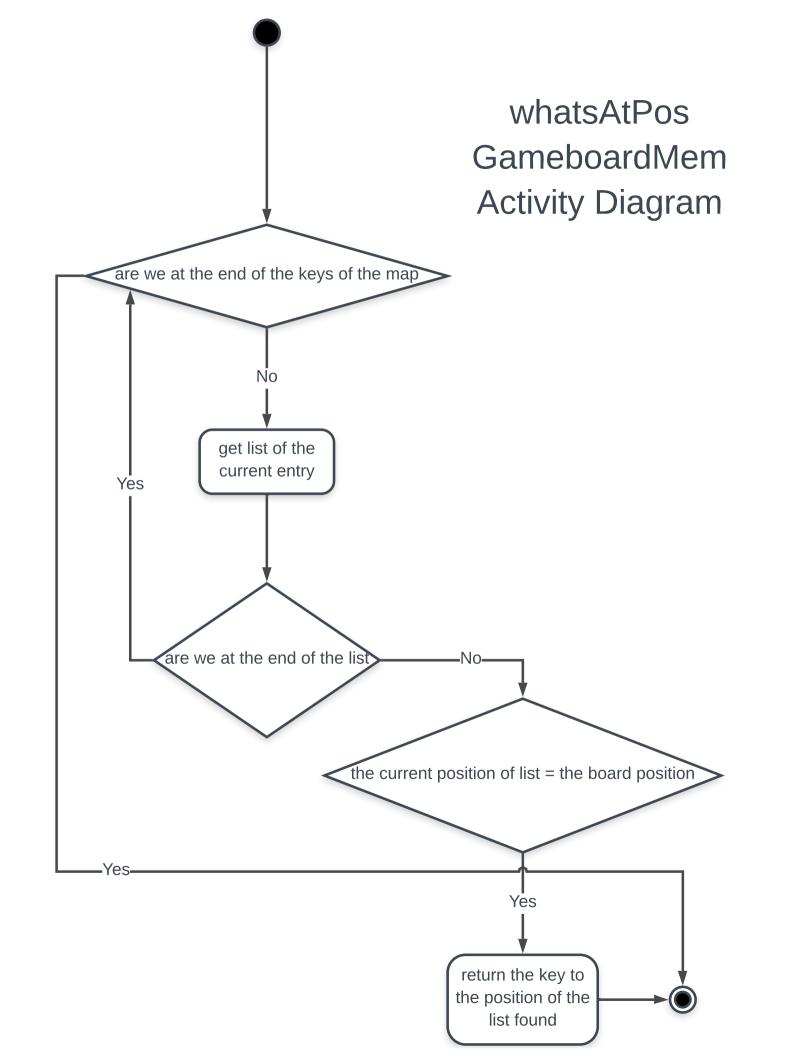


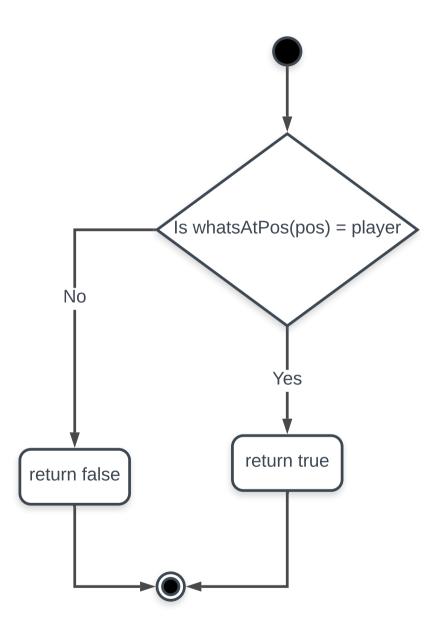
checkDiagonalWin Activity Diagram



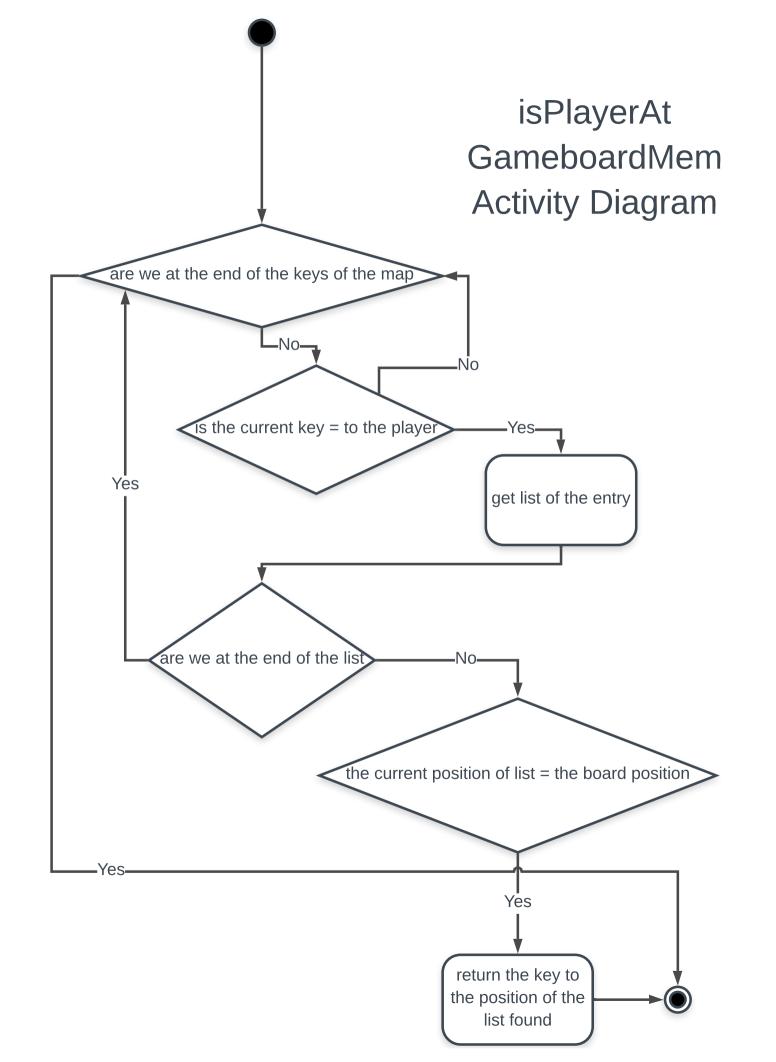


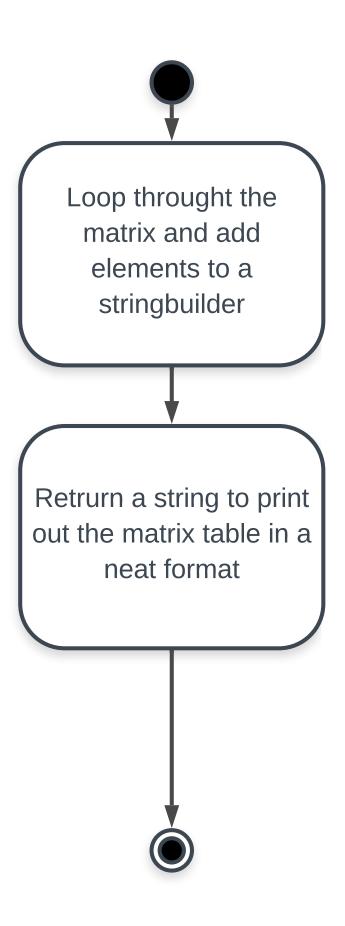
whatsAtPos Activity Diagram





isPlayerAt Activity Diagram





toString Activity Diagram

Requirements Analysis

Functional Requirements

- As a user, I can place my marker on the board, so I can try to win
- As a user, I can put my marker in a empty spot, so as not to erase someone elses marker.
- As a user, I can pick a row and column within the bounds of the matrix, so as to not go out of bounds
- As a user, I can place my markers in a horizontal pattern, so I can win horizontally
- As a user, I can place my markers in a vertical pattern, so I can win vertically
- As a user, I can place my markers in a diagonal pattern from top-left to bottom-right, so I can win diagonally.
- As a user, I can place my markers in a diagonal pattern from bottom-left to top-right, so I can win diagonally
- As a user, I can choose to play a new game, so I can play over again from the beginning
- As a user, I can have my game end in a tie, so neither me or my opponent win
- As a user, I can choose the number of rows, so I can dictate the amount of rows I want
- As a user, I can choose the number of columns, so I can dictate the amount of columns I want
- As a user, I can choose how many I need in a row to win, so I can dictate the number in row to win
- As a user, I can choose how many players that I can have, so I can play with multiple people

Non-Functional Requirments

- system must work in intellij
- must be written in c++
- must have at least 2 players
- can have a max of 10 characters
- 0,0 is the top left of the board
- the game is written in Java
- the game has a user interface

Test Documentation

void GameBoard(int rows, int cols, int num_to_win)

Inpu	t			(Output	Reason:
State	e: 0	1	2	9	State of board is unchanged	The reason this is a distinct case is because the constructor is tested constructing the board with the
0						minimum size.
2	1 2					Function Name: test_Constructor_min_game_board_size
rows:	_					
cols=						
num_	_to_\	win=	3			

void GameBoard(int rows, int cols, int num_to_win)

· · · · · · · · · · · · · · · · · · ·		
Input	Output	Reason:
State:	State of board is unchanged	The reason this is a distinct case is because we test
100x100 board with nothing		the constructor with a board that is the maximum
in it		size
rows=100		Function Name:
cols=100		test_Constructor_max_game_board_size
num_to_win=3		

void GameBoard(int rows, int cols, int num_to_win)

Inpu State						Output State of board is unchanged	Reason: The reason this test case is unique is because we
0	0	1	2	3	4	, and the second	test constructing a board with an uneven number of rows and columns
1							
2							Function Name: test_Constructor_uneven_game_board_size
rows=3 cols=5 num_to_win=3							

boolean checkSpace(BoardPosition pos)

Input	t					Output	Reason:
State	e:					checkSpace = true	The reason this test case is unique is because we are
	0	1	2	3	4		checking a space that is empty
0						State of board is unchanged	
1							Function Name:
2							test_checkSpace_empty_space
3							
4	4						
		_					
pos.g							
pos.g	getCo	olum	n=4				

boolean checkSpace(BoardPosition pos)

Inpu						Output checkSpace = false	Reason: The reason this test case is unique is because we are
	0	1	2	3	4	•	testing a space that already has something in it
0						State of board is unchanged	
1							Function Name:
2							test_checkSpace_space_already_has_character_in_it
3		Х					
4							
pos.	-						

boolean checkSpace(BoardPosition pos)

Inpu	Input					Output	Reason:
State	e:					checkSpace = false	The reason this test case is unique is because we
	0	1	2	3	4		are testing a space that is out of bounds
0						State of board is unchanged	
1							Function Name:
2	2						test_checkSpace_space_out_of_bounds
3							
4							
	pos.getRow=6						
pos.g	-						

boolean checkHorizontalWin(BoardPosition lastPos, char

Inpu	t					Output	Reason:
State	State: (number to win = 3)					checkHorizontalWin = true	The reason this test case is unique is because we
	0 1 2 3 4			3	4		are checking a horizontal win in the top left
0	Х	Х	Х			State of board is unchanged	corner of the board which has to check all the
1							way to the minimum column number.
2							
3							Function Name:
4							test_checkHorizontalWin_top_left_corner
pos.getRow=0							
pos.getColumn=2							
play	er=X						

$boolean\ check Horizontal Win (Board Position\ last Pos,\ char$

Inpu	t					Output	Reason:
State	e: (nu	ımbe	r to v	vin =	= 3)	checkHorizontalWin = true	The reason this test case is unique is because we
	0	1	2	3	4		are testing for a win in the middle of the board
0						State of board is unchanged	where we don't have to check all the way to the
1							edges.
2		Х	Х	Χ			Function Name
3							Function Name: test_checkHorizontalWin_middle_of_board
4							test_checkHohzontalvviii_illiddie_ol_boald
pos.	getRo	ow=2					
pos.	getCo	olum	n=3				
play	er=X						

$boolean\ check Horizontal Win (Board Position\ last Pos,\ char$

Inpu	t					Output	Reason:
State	e: (nu	ımbe	r to v	vin =	: 3)	checkHorizontalWin = true	The reason this test case is unique is because we are
	0	1	2	3	4		testing for a win in the bottom right corner where we are
0						State of board is unchanged	checking all the way to the maximum column size.
1							
2	2						Function Name:
3							test_checkHorizontalWin_bottom_right_of_board
4			Χ	Χ	Х		
pos.	pos.getRow=4						
pos.	getCo	olum	n=4				
playe	er=X						

boolean checkHorizontalWin(BoardPosition lastPos, char

Inpu	t					Output	Reason:
State	State: (number to win = 3)					checkHorizontalWin = true	The reason this test case is unique is because we are testing
	0	1	2	3	4		for a win in were a marker is placed in front of two of the
0						State of board is unchanged	same markers so it doesn't check left and it only has to
1							check right.
2		Х	Х	Х			
3							Function Name:
4							test_checkHorizontalWin_place_at_beginning_of_2markers
pos.	pos.getRow=2 pos.getColumn=1 player=X						

boolean checkHorizontalWin(BoardPosition lastPos, char

Inpu	t					Output	Reason:
State	e: (nu	ımbe	r to v	win =	= 3)	checkHorizontalWin = true	The reason this test case is unique is because we are testing
	0 1 2 3 4		4		for a win in were a marker is placed in the middle of 2 of		
0						State of board is unchanged	the same markers so it has to check left and right.
1							
2		Х	Х	Х			Function Name:
3							test_checkHorizontalWin_place_middle_of_2markers
4							
pos.g pos.g playe	getCo				,		

boolean checkVerticalWin(BoardPosition lastPos, char

Inpu	t					Output	Reason:
State	e: (nu	mbe	r to ۱	win =	: 3)	checkVerticalWin = true	The reason this test case is unique is because we are
	0	1	2	3	4		checking a vertical win in the top left corner of the board
0	Χ					State of board is unchanged	which has to check all the way to the minimum row
1	Χ						number.
2	Χ						Function Name:
3							test_checkVerticalWin_top_left_corner
4							test_checkverticalvviii_top_left_corner
pos.	pos.getRow=2						
pos.g	getCo	lumi	n=0				
playe	er=X						

$boolean\ check Vertical Win (Board Position\ last Pos,\ char$

Inpu	t					Output	Reason:
State	e: (nu	ımbe	rto١	vin =	: 3)	checkVerticalWin = true	The reason this test case is unique is because we are
	0 1 2 3 4			3	4		testing for a win in the middle of the board where we
0						State of board is unchanged	don't have to check all the way to the edges.
1			Χ				
2			Х				Function Name:
3			Χ				test_checkVerticalWin_middle_of_board
4							
pos.	getRo	w=3					
pos.	getCo	olum	n=2				
playe	er=X						

boolean checkVerticalWin(BoardPosition lastPos, char

Inpu State		ımbe	r to v	win =	: 3)	Output checkVerticalWin = true	Reason: The reason this test case is unique is because we are
	0 1 2 3 4			3	4	State of board is unchanged	testing for a win in the bottom right corner where we are checking all the way to the maximum row size.
0						State of board is dischariged	checking all the way to the maximum row size.
							Function Name:
2					Х		test checkVerticalWin bottom right of board
3					Х		test_eneckverticalvviii_bottom_right_or_board
4					Х		
pos.	pos.getRow=4						
pos.	getCo	olum	n=4				
play	er=X						

boolean checkVerticalWin(BoardPosition lastPos, char

Inpu	t					Output	Reason:
State	e: (nu	mbe	r to ۱	vin =	: 3)	checkVerticalWin = true	The reason this test case is unique is because we are
	0	1	2	3	4		testing for a win in were a marker is placed in front of two
0						State of board is unchanged	of the same markers so it doesn't check up and it only has
1		Χ					to check down.
2	X						Function Name:
3		Χ					test_checkVerticalWin_place_at_beginning_of_2markers
4							test_checkverticatvviii_place_at_begiiiiiiiig_oi_ziiiaikeis
pos.g	getRo	w=1	,				
pos.	getCo	lum	n=1				
playe	er=X						

$boolean\ check Vertical Win (Board Position\ last Pos,\ char$

Inpu	t					Output	Reason:
State	e: (nu	ımbe	rto١	win =	: 3)	checkVerticalWin = true	The reason this test case is unique is because we are
	0 1 2 3 4			3	4		testing for a win in were a marker is placed in the middle
0						State of board is unchanged	of 2 of the same markers so it has to check up and down.
1		Х					
2		Х					Function Name:
3		Х					test_checkVerticalWin_place_middle_of_2markers
4							
pos.	getRo	w=2					
pos.	getCo	olum	n=1				
playe	er=X						

boolean checkDiagonalWin(BoardPosition lastPos, char player)

Inpu	t					Output	Reason:				
State	e: (nu	ımbe	rto١	vin =	3)	checkDiagonalWin =	This test case is unique because we are checking the outer				
	0	1	2	3	4	true	bounds of the bottom left of the board. We will have to check all				
0							the way to the max row number and the minimum column				
1						State of board is	number				
2			Х			unchanged					
3		Х					Function Name:				
4	Х						test_checkDiagonalWin_bottomleft_to_topright_at_bottom_left				
pos.	pos.getRow=2 pos.getColumn=2 player=X										

boolean checkDiagonalWin(BoardPosition lastPos, char player)

Inpu	t					Output	Reason:
State	e: (nu	ımbe	r to v	win =	= 3)	checkDiagonalWin = true	The test case is unique because we are checking the outer
	0 1 2 3 4			3	4		bounds of the top right of the board. We have to check all the
0					Х	State of board is	way to the max column number and minimum row number.
1	1 X					unchanged	
2	2 X						Function Name:
3							test_checkDiagonalWin_bottomleft_to_topright_at_top_right
4							
pos.	getRo	ow=0)				
pos.	getCo	olum	n=4				
play	er=X						

boolean checkDiagonalWin(BoardPosition lastPos, char player)

Input	t					Output	Reason:
State	State: (number to win = 3)					checkDiagonalWin = true	This test case is unique because we are testing the outer
	0	1	2	3	4		bounds of bottom right of the board. We have to check all the
0						State of board is	way to the max row and column number.
1						unchanged	
2			Х				Function Name:
3				Х			test_checkDiagonalWin_topleft_to_bottomright_bottom_right
4					Х		
pos.getRow=4							
pos.g	getCo	lum	n=4				
playe	er=X						

boolean checkDiagonalWin(BoardPosition lastPos, char player)

Inpu	t					Output	Reason:
State	e: (nu	ımbe	rto۱	win =	= 3)	checkDiagonalWin = true	This test case is unique because we are checking the outer
	0 1 2 3 4			3	4		bounds of the top left of the board. We have to check all
0	Х					State of board is unchanged	the way to the minimum row and column number.
1	. X						
2	X						Function Name:
3							test_checkDiagonalWin_topleft_to_bottomright_top_left
4							
pos.	getRo	ow=0					
pos.	getCo	olum	n=0				
play	er=X						

boolean checkDiagonalWin(BoardPosition lastPos, char player)

Inpu	t					Output	Reason:
State	e: (nu	ımbe	rto١	vin =	: 3)	checkDiagonalWin = true	This test case is unique because we are checking for a win
	0 1 2 3 4			3	4		in a smaller part of the board at the bottom left. We aren't
0						State of board is unchanged	checking for a win in the larger diagonal areas.
1							
2	Х						Function Name:
3		Х					test_checkDiagonalWin_bottom_left_small_diagonal
4			Х				
pos.	pos.getRow=4						
pos.	getCo	olum	n=2				
play	er=X						

boolean checkDiagonalWin(BoardPosition lastPos, char player)

Inpu	t					Output	Reason:
State	e: (nu	ımbe	rto۱	vin =	3)	checkDiagonalWin = true	This test case is unique because we are checking for a win
	0 1 2 3 4			3	4		in a smaller part of the board at the bottom right and we
0						State of board is unchanged	are going from bottom left to top right.
1							
2					Х		Function Name:
3				Χ			test_checkDiagonalWin_bottom_right_small_diagonal
4			Х				
pos.	getRo	ow=2					
pos.	getCo	olum	n=4				
playe	er=X						

boolean checkDiagonalWin(BoardPosition lastPos, char player)

Inpu	ıt					Output	Reason:
Stat	e: (nı	ımbe	rto١	vin =	: 3)	checkDiagonalWin = true	This test case is unique because the last marker we place is
	0	0 1 2 3 4			4		in the middle of two of the same markers so we have to
0	Х					State of board is unchanged	check both directions of the diagonal in the top left of the
1		Х					board. This is an edge case.
2	2 X						Function Name
3							Function Name: test_checkDiagonalWin_top_left_place_marker_in_middle
4							test_checkblagonalvviii_top_left_place_marker_iii_middle
pos.	getRo	ow=1					
pos.	getCo	olum	n=1				
play	er=X						

boolean checkDiagonalWin(BoardPosition lastPos, char player)

Inpu	t					Output	Reason:		
State	e: (nu	ımbe	r to v	win =	= 3)	checkDiagonalWin = true	This test case is unique because we are placing the last		
	0 1 2 3 4			3	4		marker in the middle of two of the same markers in the top		
0					Х	State of board is	right of the board. We will have to check in both directions.		
1				Х		unchanged	Where we are checking for the win is also why this test case		
2			Х				is unique.		
3							Function Name:		
4							test_checkDiagonalWin_top_right_place_marker_in_middl		
pos.	getRo)w=1		ı			test_checkblagonalwin_top_right_place_marker_in_middle		
pos.	getCo	olum	n=3						
play	er=X								

boolean checkForDrawWin()

Input					Output	Reason:		
State:	(nu	mbe	r to v	win = 3)	checkForDraw = true	This test case is unique because we are checking for a		
0 1 2			2			draw when the board is full and there is not a winner.		
0	0 X O X			State of board is unchanged				
1	0	Χ	0			Function Name:		
2	2 0 X 0				test_checkForDraw_board_is_full_without_win			
	2 0 1 1 0							

boolean checkForDrawWin()

Inpu	t				Output	Reason:			
State	e: (nu	mbe	r to v	win = 3)	checkForDraw = false	This test case is unique because we are testing to make			
	0 1 2					sure after we place just one marker that we are not told			
0	0 X		1	State of board is unchanged	that there is a draw.				
1	1								
2						Function Name:			
						test_checkForDraw_board_place_only_one_marker			

boolean checkForDrawWin()

Input	Output	Reason:		
State: (number to win = 3)	checkForDraw = false	This test case is unique because we are testing to make		
		sure that even though the board is one spot away from		
0 X O X	State of board is unchanged	being full without a winner that we do not get told that		
1 O X O		there was a draw.		
2 O X		Function Name: test_checkForDraw_board_place_only_one_marker		

boolean checkForDrawWin()

Inpu	t				Output	Reason:
State	e: (nu	ımbe	r to	win = 3)	checkForDraw = false	This test case is unique because we are testing to make
	0 1 2					sure that even though the board is half full that we are not
0	0 X O X			State of board is unchanged	told that there is a draw	
1	1 O X				E coltan Nove	
2						Function Name:
				-		test_checkForDraw_board_place_only_one_marker

char whatsAtPos(BoardPosition pos)

Input	t					Output	Reason:			
State:						whatsAtPos= empty space	This test case is unique because we are testing to make			
	0	1	2	3	4		sure that whats AtPos will return an empty space if there is $% \left\{ 1,2,\ldots ,n\right\}$			
0						State of board is unchanged	not anything in that position.			
1										
2						Function Name:				
3							test_whatsAtPos_empty_space			
4										
			<u> </u>							
pos.getRow=2										
pos.getColumn=2										
poolgood man =										

char whatsAtPos(BoardPosition pos)

Inpu	t					Output	Reason:
State	State:					whatsAtPos= X	The reason this test case is unique is because we are
	0	1	2	3	4		testing to see that whatsAtPos will return the correct
0						State of board is unchanged	marker is it's in the middle of board.
1							Function Name:
2			Х				Function Name:
3							test_whatsAtPos_marker_in_middle_of_board
4							
pos.	-						

char whatsAtPos(BoardPosition pos)

Input						Output	Reason:			
State:						whatsAtPos= X	The reason this test case is unique is because we are			
	0	1	2	3	4		testing to that whatsAtPos will return the correct marker if			
0	Х					State of board is unchanged	the marker is in the top left corner. It is at the edge of the			
1							board so that is why it unique.			
2										
3							Function Name:			
4							test_whatsAtPos_marker_top_left_corner			
4										
	pos.getRow=0 pos.getColumn=0									

char whatsAtPos(BoardPosition pos)

Input State:						Output whatsAtPos= X	Reason: The reason this test case is unique is because we are			
0 1 0 1		1	2	3	4 X	State of board is unchanged	testing to that whatsAtPos will return the correct marker if the marker is in the top right corner. It is at the edge of the board so that is why it unique.			
3 4	3				Function Name: test_whatsAtPos_marker_top_right_corner					
pos.getRow=0 pos.getColumn=4										

char whatsAtPos(BoardPosition pos)

	Input State:					Output whatsAtPos= X	Reason: The reason this test case is unique is because we are			
0	0 1 2 3 4		4	State of board is unchanged	testing to that whatsAtPos will return the correct marker if the marker is in the bottom left corner. It is at the edge of					
2							the board so that is why it unique.			
3	3				Function Name: test_whatsAtPos_marker_bottom_left_corner					
	pos.getRow=4 pos.getColumn=0									

char whatsAtPos(BoardPosition pos)

Input State:						Output whatsAtPos= X	Reason: The reason this test case is unique is because we are
	0 1 2 3 4		4		testing to that whatsAtPos will return the correct marker if		
0						State of board is unchanged	the marker is in the bottom right corner. It is at the edge
1							of the board so that is why it unique.
2							Function Name
3	3			Function Name: test whatsAtPos marker bottom right corner			
4					Х		test_whatsAtFos_marker_bottom_right_comer
pos.getRow=4 pos.getColumn=0							

char whatsAtPos(BoardPosition pos)

Input State:						Output whatsAtPos= O	Reason: The reason this test case is unique is because we are			
	0 1 2 3 4		4	State of board is unchanged	testing to that whatsAtPos will return the correct marker if					
0			0			State of board is unchanged	there are different markers on the board.			
1							Franking Names			
2							Function Name:			
3	- 				test_whatsAtPos_O_Marker					
4										
pos.getRow=0 pos.getColumn=2					· · · · ·					

void placeMarker(BoardPosition pos, char player)

	ora pracernariter (Boarar osit									
Inp	Input									
Sta	State:									
	0	1	2	3	4					
0										
1										
2	Х									
3										
4	Х				Χ					

player=O pos.getRow=2 pos.getColumn=2

Output

State	State:						
	0	1	2	3	4		
0							
1							
2	Χ		0				
3							
4	Χ			Χ			

Reason:

The reason this test case is unique is because we are placing a marker which has not been placed on the board yet

Function Name:

test_placeMarker_place_new_player

void placeMarker(BoardPosition pos, char player)

Input								
State	State:							
	0	1	2	3	4			
0								
1								
2								
3								
4								

player=X pos.getRow=0 pos.getColumn=0

Output State:

	0	1	2	3	4
0	Χ				
1					
2					
თ					
4					

Reason:

The reason this test case is unique is because we are testing placing a marker in the top left corner of the board which is an edge case.

Function Name:

test_placeMarker_place_top_left_corner

void placeMarker(BoardPosition pos, char player)

Input								
Stat	State:							
	0	1	2	3	4			
0								
1								
2								
3								
4								

player=X pos.getRow=4 pos.getColumn=0

Output

State:

	0	1	2	3	4
0					
1					
2					
3					
4	Χ				

Reason:

The reason this test case is unique is because we are testing placing a marker in the bottom left corner which is an edge case.

Function Name:

test_placeMarker_place_bottom_left_corner

void placeMarker(BoardPosition pos, char player)

Inpu	Input State:						
State	<u>:</u> :						
	0	1	2	3	4		
0							
1							
2							
3							
4							

player=X pos.getRow=4 pos.getColumn=4

Output

State:

	0	1	2	3	4
0					
1					
2					
3					
4					Χ

Reason:

The reason this test case is unique is because we are testing placing a marker in the bottom right corner of the board which is an edge case.

Function Name:

test_placeMarker_place_bottom_right_corner

void placeMarker(BoardPosition pos, char player)

Input State:

	0	1	2	3	4
0					
1					
2					
3					
4					

player=X pos.getRow=0 pos.getColumn=4

Output

State:

	0	1	2	3	4	
0					Χ	
1						
2						
3						
4						

Reason:

The reason this test case is unique is because we are testing placing a marker in the top right corner of the board which is an edge case.

Function Name:

test_placeMarker_place_top_right_corner