GameBoard(int nR, int nC, int nW)

Inputs		Expecte	d Output	t	Reason for Test Case		
State:			ite:		This case represents the lower		
board = null		boa	rd =		bound of a GameBoard's size. In		
numRow = null		0	1	2	this case, numToWin, numCol,		
numCol = null	0				and numRow are all equal to		
numToWin = null	1				their lowest possible values		
nR = MIN_ROW	2				when the object is created.		
nC = MIN_COLUMNS		numR	ow = 3				
nW = MIN_TO_WIN		numC	Col = 3				
		numTo	Win = 3				

GameBoard(int nR, int nC, int nW)

Inputs		Exp	ecte	d Out	put		Reason for Test Case
State: board = null numRow = null numCol = null numToWin = null nR = MAX_ROWS nC = MAX_COLUMNS nW = 30	0 1 2 99	0	Sta	te: rd = 2		99	This case represents the upper bound of a GameBoard's size. In this case, the number of rows and columns are set their maximum values but numToWin is set to a different, smaller value.
		n	umCo mToV	ol = 10	0		

GameBoard(int nR, int nC, int nW)

Inputs	Expected Output								Reason for Test Case				
State:	State:								This case represents the				
board = null				bo	ard	=				general case for a GameBoard.			
numRow = null		0	1	2	3	4	5	6	7	It is a rectangular shape where			
numCol = null	0									the numToWin is not at either			
numToWin = null	1									extreme, so all three inputs are			
nR = 10	2									different from each other.			
nC = 8	3									Most of the available layouts			
nW = 4	4									for a new board will be like this			
	5									one.			
	6												
	7												
	8												
	9												
	numRow = 10												
			r	num	Col	= 8							
			nu	mΤ	oWi	n =	4						

GameBoard(int nR, int nC, int nW)

Inputs	Expected Output	Reason for Test Case
State: board = null numRow = null numCol = null numToWin = null nR = MAX_ROWS nC = MIN_COLUMNS nW = MIN_TO_WIN	State: board = 0 1 2 0 1 2 0 1 2 0	This case represents a wildly disproportionate GameBoard. numRow and numCol are set to their OPPOSITE extremes, and numToWin is set to its minimum. This tests the case where a board is as long or wide as possible.
	numRow = 100 numCol = 3 numToWin = 3	

Boolean checkSpace(BoardPosition pos)

			lr	nput	:				Expected Outputs	Reason for Test Case
			St	tate	:					This tests the case where an
	0	1	2	3	4	5	6	7	The state of the board is	invalid (out of range) position
0									unchanged	is passed in for evaluation.
1		Χ								
2			0						checkSpace = false	
3										
4					Χ					
5										
6										
7										
8										
9										
	Pos.getRow() = -5									
	pos.getColumn() = 15									

Boolean checkSpace(BoardPosition pos)

				npı	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					This tests the case where a
	0	1	2	3	4	5	6	7	The state of the board is	position is checked that has
0									unchanged	already been played in. This is
1		Χ								the case where a potentially
2			0						checkSpace = false	valid input has been passed in,
3										but it has already been used
4					Χ					once before and will not be
5										allowed again.
6										
7										
8										
9										
	Pos.getRow = 2									
	Pos.getRow = 2									

Boolean checkSpace(BoardPosition pos)

			lr	put					Expected Outputs	Reason for Test Case
			St	ate	:					This tests the case where the
	0	1	2	თ	4	5	6	7	The state of the board is	method is called on an
0									unchanged	unoccupied position in range.
1		Χ								This is the case where the
2			0						checkSpace = true	position checked is valid.
3										
4					Χ					
5										
6										
7										
8										
9										
		Pos	.get	Rov	v() =	= 6				
	Pos.getColumn() = 2									

			ı	npu	ıt				Expected Outputs	Reason for Test Case
			5	State	e:					This tests the case where
	0	1	2	3	4	5	6	7	The state of the board is	lastPos is less than numToWin
0	Χ							0	unchanged	positions away from the left
1	Χ	Χ	Χ	Χ				0		side bound of the game
2			Χ			0			checkHorizontalWin = true	board.
3				Χ	0		0	0		
4										
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	lastPos.getRow() = 1 lastPos.getColumn() = 1 player = X numToWin = 4									

Boolean checkHorizontalWin(BoardPosition lastPos, char player)

			-	npu	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					This tests the case where
	0	1	2	3	4	5	6	7	The state of the board is	lastPos is less than numToWin
0	Χ							0	unchanged	positions away from the right
1	Χ		Χ	Χ				0		side bound of the game
2			Χ			0			checkHorizontalWin = true	board.
3				Χ	0	0	0	0		
4										
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	lastPos.getRow() = 3									
	lastPos.getColumn() = 5									
	player = O									
	numToWin = 4									

				npu	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					This case tests for when a
	0	1	2	3	4	5	6	7	The state of the board is	horizontal victory is achieved
0	Χ							0	unchanged	by aligning more than the
1	Χ	Χ	Χ	Χ	Χ			0		required number of game
2			Χ			0			checkHorizontalWin = true	pieces.
3				Χ	0		0	0		
4										
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	lastPos.getRow() = 1									
	lastPos.getColumn() = 1									
	player = X									
		n	um	ΓοΨ	/in =	4				

Boolean checkHorizontalWin(BoardPosition lastPos, char player)

			I	npu	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					This tests the case where a
	0	1	2	3	4	5	6	7	The state of the board is	player has placed their marker
0	Χ							0	unchanged	in a line of their own markers,
1	Χ		Χ	Χ				0		but there are not enough
2			Χ			0			checkHorizontalWin = false	markers aligned to achieve
3				Χ	0		0	0		victory.
4										
5										
6				Χ	0	0	0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	lastPos.getRow() = 6									
	lastPos.getColumn() = 5									
	player = O									
		n	um7	ΓοΨ	/in =	4				

			I	npu	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					This tests the case where
	0	1	2	3	4	5	6	7	The state of the board is	lastPos is less than numToWin
0	Χ							0	unchanged	positions away from the
1	Χ		Χ	Χ				0		upper bound of the game
2			Χ			0		0	checkVerticalWin = true	board.
3				Χ	0		0	0		
4										
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	lastPos.getRow() = 2 lastPos.getColumn() = 7 player = 0 numToWin = 4									

Boolean checkVerticalWin(BoardPosition lastPos, char player)

			I	npu	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					This tests the case where
	0	1	2	3	4	5	6	7	The state of the board is	lastPos is less than numToWin
0	Χ							0	unchanged	positions away from the lower
1	Χ		Χ	Χ				0		bound of the game board.
2			Χ			0			checkVerticalWin = true	
3				Χ	0		0	0		
4										
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0				0			
	lastPos.getRow() = 9 lastPos.getColumn() = 6 player = 0 numToWin = 4									

			I	npu	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					This tests the case where the
	0	1	2	3	4	5	6	7	The state of the board is	player has aligned some of
0	Χ							0	unchanged	their markers vertically but
1	Χ		Χ	Χ				0		identifies that they have failed
2			Χ			0			checkVerticalWin = false	to align enough to win.
3				Χ	0		0	0		
4							0			
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	lastPos.getRow() = 4						4			
	lastPos.getColumn() = 6						- 6			
	player = O									
		n	um	ΓοΨ	/in =	4				

Boolean checkVerticalWin(BoardPosition lastPos, char player)

			I	npu	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					This tests the case where the
	0	1	2	3	4	5	6	7	The state of the board is	player has won by aligned
0	Χ							0	unchanged	more than the required
1	Χ		Χ	Χ				0		number of game pieces in a
2			Χ			0			checkVerticalWin = true	vertical line.
3				Χ	0		0	0		
4							0			
5							0			
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	lastPos.getRow() = 5 lastPos.getColumn() = 6 player = 0 numToWin = 4									

			I	Inpu	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					Tests the case where the user
	0	1	2	3	4	5	6	7	The state of the board is	wins along the minor
0	Χ							0	unchanged	diagonal, but last pos is at
1	Χ		Χ	Χ			0	0		least numToWin game pieces
2			Χ			0			checkDiagonalWin = true	from the top right corner of
3				Χ	0		0	0		the board.
4										
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	lastPos.getRow() = 1						1			
	lastPos.getColumn() = 6									
			pla	yer	= O					
	numToWin = 4									

Boolean checkDiagonalWin(BoardPosition lastPos, char player)

				npu	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					Tests the case where the user
	0	1	2	3	4	5	6	7	The state of the board is	wins along the minor diagonal,
0	Χ							0	unchanged	but last pos is at least
1	Χ		Χ	Χ				0		numToWin game pieces from
2			Χ			0			checkDiagonalWin = true	the bottom right corner of the
3				Χ	0		0	0		board.
4										
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0					0		
	lastPos.getRow() = 9 lastPos.getColumn() = 7 player = 0 numToWin = 4									

				npu	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					Tests the case where the user
	0	1	2	თ	4	5	6	7	The state of the board is	wins along the minor diagonal,
0	Χ							0	unchanged	but last pos is at least
1	Χ	Χ	Χ	Χ				0		numToWin game pieces from
2			Χ			0			checkDiagonalWin = true	the top left corner of the
3				Χ	0		0	0		board.
4										
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	lastPos.getRow() = 1 lastPos.getColumn() = 1 player = X numToWin = 4									

Boolean checkDiagonalWin(BoardPosition lastPos, char player)

				npı	ıt				Expected Outputs	Reason for Test Case
			9	State	e:					Tests the case where the user
	0	1	2	3	4	5	6	7	The state of the board is	wins along the minor diagonal,
0	Χ							0	unchanged	but last pos is at least
1	Χ		Χ	Χ				0		numToWin game pieces from
2			Χ			0			checkDiagonalWin = true	the bottom left corner of the
3				Χ	0		0	0		board.
4										
5										
6				Χ	0		0			
7	0		Χ			0	0			
8		Χ					0			
9	Χ		0							
	lastPos.getRow() = 8 lastPos.getColumn() = 1 player = X numToWin = 4									

			ı	npu	t				Expected Outputs	Reason for Test Case
			S	state	e:					This tests the case where the
	0	1	2	3	4	5	6	7	The state of the board is	lastPosition is in a line of
0	Χ							0	unchanged	characters that borders the
1	Χ		Χ	Χ				0		bound of the game board at
2			Χ			0			checkDiagonalWin = false	two points.
3				Χ	0		0	0		
4										
5										
6				Χ	0		0			
7	0		Χ			0	0			
8		0					0			
9	Χ		0							
	lastPos.getRow() = 8 lastPos.getColumn() = 1 player = 0 numToWin = 4									

Boolean checkDiagonalWin(BoardPosition lastPos, char player)

			I	npu	ıt				Expected Outputs	Reason for Test Case
			5	State	e:					Tests the case where the
	0	1	2	3	4	5	6	7	The state of the board is	lastPos is in a line of
0	Χ							0	unchanged	characters along the main
1	Χ		Χ	Χ				0		diagonal, but there are not
2			Χ			0			checkDiagonalWin = false	enough markers in line to
3				Χ	0		0	0		achieve a victory.
4					Χ					
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	lastPos.getRow() = 4						4			
	lastPos.getColumn() = 4						= 4			
			pla	ıyer	= X					
		n	um1	ΓοW	/in =	4				

				Inpu	ıt				Expected Outputs	Reason for Test Case
				State					, , ,	Tests the case where the
	0	1	2	3	4	5	6	7	The state of the board is	player wins by aligning more
0	Χ							0	unchanged	than the required number of
1	Χ	Χ	Χ	Χ				0		markers along the main
2			Χ			0			checkDiagonalWin = true	diagonal
3				Χ	0		0	0		
4					Χ					
5										
6				Χ	0		0			
7	0		Χ			0	0			
8							0			
9	Χ		0							
	<pre>lastPos.getRow() = 1 lastPos.getColumn() = 1 player = X numToWin = 4</pre>									

Boolean checkDraw(void)

			I	npu	t				Expected Outputs	Reason for Test Case
			S	tate	2:					This case represents the
	0	1	2	3	4	5	6	7	The state of the board is	situation where a draw has
0	Α	В	С	D	Ε	F	G	Н	unchanged	resulted. There is no winner
1	1	J	Α	В	С	D	Ε	F		
2	G	Н	I	J	Α	В	С	D	checkDraw = true	
3	Ε	F	G	Н	ı	J	Α	В		
4	С	D	Ε	F	G	Н	I	J		
5	Α	В	С	D	Ε	F	G	Н		
6	1	J	Α	В	С	D	Ε	F		
7	G	Н	-	J	Α	В	С	D		
8	Ε	F	G	Н	ı	J	Α	В		
9	С	D	Ε	F	G	Н	I	J		
				•	•					

Boolean checkDraw(void)

			lı	npu	t				Expected Outputs	Reason for Test Case
			S	tate	::					This case represents a case
	0	1	2	3	4	5	6	7	The state of the board is	where no draw is present and
0	Χ								unchanged	some players have already
1					0					made moves on the board.
2									checkDraw = false	
3					0					
4			Χ		0					
5				Χ						
6										
7										
8										
9										

Boolean checkDraw(void)

			Ir	nput	:				Expected Outputs	Reason for Test Case
			St	tate	:					This case represents a situation
	0	1	2	3	4	5	6	7	The state of the board is	when the method is called on
0									unchanged	an empty board where no
1										players have made a move.
2									checkDraw = false	
3										
4										
5										
6										
7										
8										
9										

Boolean checkDraw(void)

			I	npu	t				Expected Outputs	Reason for Test Case
	State: 0 1 2 3 4 5 6 7									This case represents a
	0	1	2	3	4	5	6	7	The state of the board is	situation where there is a
0	Χ	Α	В	С	D	Ε	F	G	unchanged	draw, but a win condition is
1	Н	Χ	ı	Α	В	С	D	Ε		present on the board. This is
2	F	G	Χ	Н	ı	Α	В	С	checkDraw = true	to illustrate that the method
3	D	Ε	F	Χ	G	Н	ı	Α		should return true when the
4	В	С	D	Ε	Χ	F	G	Н		board is full, even if the game
5	ı	Α	В	С	D	Χ	Ε	F		has not truly ended in a draw.
6	G	Н	ı	Α	В	С	Χ	D		
7	Ε	F	G	Н	ı	Α	В	Х		
8	С	D	Ε	F	G	Н	ı	Α		
9	В	С	D	Ε	F	G	Н	ı		
	numToWin = 4									
		П	uilli	UVV	111 =	4				

char whatsAtPos(BoardPosition pos)

				Inpu	ıt				Expected Outputs	Reason for Test Case
	State: 0 1 2 3 4 5 6 7									This tests the case where a
	0	1	2	3	4	5	6	7	The state of the board is	position with a valid row is
0									unchanged	passed in, but its column over
1										the allowed maximum.
2			Χ				0		whatsAtPos = ' '	
3										
4				Χ	0					
5						0				
6			Χ							
7										
8										
9										
		_				_				
	Pos.getRow = 5 Pos.getColumn = 15						-			
		rus	.get	COIL	attiti	- 13)			

char whatsAtPos(BoardPosition pos)

				Inpu	ıt				Expected Outputs	Reason for Test Case
				Stat	e:					This tests the case where the
	0	1	2	3	4	5	6	7	The state of the board is	position is empty and in range.
0									unchanged	
1										
2			Χ				0		whatsAtPos = ' '	
3										
4				Χ	0					
5						0				
6			Χ							
7										
8										
9										
	Pos.getRow = 0									
		Pos	.ge	tCol	umr	ı = 0				

char whatsAtPos(BoardPosition pos)

				Inpu	ıt				Expected Outputs	Reason for Test Case
			,	Stat	e:					This case represents a situation
	0	1	2	3	4	5	6	7	The state of the board is	when a position that has
0									unchanged	negative coordinates is passed
1										in.
2			Χ				0		whatsAtPos = ' '	
3										
4				Χ	0					
5						0				
6			X							
7										
8										
9										
			_		ow = ow =					

char whatsAtPos(BoardPosition pos)

				Inpu	ıt				Expected Outputs	Reason for Test Case
			,	Stat	e:					Tests the case where a position
	0	1	2	3	4	5	6	7	The state of the board is	with a column over the
0									unchanged	allowed maximum is passed in.
1										
2			Χ				0		whatsAtPos = ' '	
3										
4				Χ	0					
5						0				
6			Х							
7										
8										
9										
			_		w = umr	: 15 n = 5				

char whatsAtPos(BoardPosition pos)

			I	npu	t				Expected Outputs	Reason for Test Case
			S	tate	9:					This case represents a
	0	1	2	3	4	5	6	7	The state of the board is	situation where a position is
0									unchanged	looked at that has a player
1										marker placed in it.
2			Χ				0		whatsAtPos = 'X'	
3										
4				Χ	0					
5						0				
6			Χ							
7										
8										
9										
			_		ow = umn					

Boolean isPlayerAtPos(BoardPosition pos, char player)

				Inpu	ıt				Expected Outputs	Reason for Test Case
			9	Stat	e:					The case represents a situation
	0	1	2	3	4	5	6	7	The state of the board is	when the method is called and
0									unchanged	to check for a player that does
1										exist on the GameBoard but in
2			Χ				0		isPlayerAtPos = false	a position that does not
3										contain that player. This checks
4				Χ	0					to make sure that the method
5						0				is not returning a positive
6			Χ							value just because a piece is
7										present on the board
8										somewhere.
9										
		P	os.g	etR	= 'X' ow = umr)			

Boolean isPlayerAtPos(BoardPosition pos, char player)

				Inpu	ıt				Expected Outputs	Reason for Test Case
				Stat	e:					This tests the situation where a
	0	1	2	3	4	5	6	7	The state of the board is	position is passed in with a
0									unchanged	non-blank character that
1										matches the data in the
2			Χ				0		isPlayerAtPos = true	GameBoard.
3										
4				Χ	0					
5						0				
6			Χ							
7										
8										
9										
			os.	getR	= 'X' low= lum					

Boolean isPlayerAtPos(BoardPosition pos, char player)

									T	T
				Inpι	ut				Expected Outputs	Reason for Test Case
			9	Stat	e:					This represents a situation
	0	1	2	3	4	5	6	7	The state of the board is	where the method is called on
0									unchanged	a taken position and given a
1										player character as input, but
2			Х				0		isPlayerAtPos = false	the input that is given does not
3										match the character that is
4				Х	0					occupying the space.
5						0				
6			Х			_				
1 7										
8										
9										
		l		l			I			
	Player = 'X'									
	Pos.getRow = 2									
	Pos.getRow = 2 Pos.getColumn = 6									

Boolean isPlayerAtPos(BoardPosition pos, char player)

				Inpu	ıt				Expected Outputs	Reason for Test Case
			Ç	State	e:					Tests the case where an
	0	1	2	3	4	5	6	7	The state of the board is	occupied space is checked to
0									unchanged	see if it is empty of not.
1										
2			Χ				0		isPlayerAtPos = false	
3										
4				Χ	0					
5						0				
6			Χ							
7										
8										
9										
	Player = ' '									
	pos.getRow = 2									
		pos	s.ge	tCol	umr	า = 2				

Boolean isPlayerAtPos(BoardPosition pos, char player)

				Inpu	ıt				Expected Outputs	Reason for Test Case
				State						This checks the situation
	0	1	2	3	4	5	6	7	The state of the board is	where an empty position is
0									unchanged	checked against a character to
1										represent an empty space.
2			Х				0		isPlayerAtPos = true	
3										
4				Χ	0					
5						0				
6			Х							
7										
8										
9										
			Pla	yer	= ' '					
	Pos.getRow = 0									
	Pos.getColumn = 0)			

void placeMarker(BoardPosition pos, char player)

				Inpu	ıt							Ex	pect	ed (Outp	outs			Reason for Test Case
			Ç	Stat	e:								Ç	Stat	e:				This tests the case where
	0	1	2	თ	4	5	6	7			0	1	2	თ	4	5	6	7	the marker is added at the
0										0								Х	first quadrant's bound.
1										1									
2			Χ				0			2			Χ				0		
3										3									
4				Χ	0					4				Χ	0				
5						0				5						0			
6	6 X												Χ						
7	7																		
8	8																		
9							9												
			Pla	yer	= 'X'	,													
		Р	os.g	etR	ow =	= 0													
Pos	.get	:Colı	umr	1 = g	etN	umC	Colui	mns											

void placeMarker(BoardPosition pos, char player)

									-										
				Inpu	ut							Ex	pect	ted (Outp	outs			Reason for Test Case
<u> </u>				Stat	e:									Stat	e:				This tests the case where
	0	1	2	3	4	5	6	7			0	1	2	3	4	5	6	7	the marker is added at the
0										0									fourth quadrant's bound.
1										1									
2			Χ				0			2			Χ				0		
3										3									
4				Х	0					4				Χ	0				
5						0				5						0			
6			Χ							6			Х						
7										7									
8										8									
9										9								Х	
		ı		ı	1	1	1	1							ı	ı	1	ı	
			Pla	yer	= 'X'	,													
F	os.	getR		•		ımR	ows	()											
	Ì	_		_	lum			.,											
			_			ns()													

void placeMarker(BoardPosition pos, char player)

				Inpu	ut							Ex	oect	ed (Outp	outs			Reason for Test Case
			(Stat	e:								(Stat	e:				This tests the case where
	0	1	2	3	4	5	6	7			0	1	2	თ	4	5	6	7	the marker is added at the
0										0	Χ								second quadrant's bound.
1										1									
2			Χ				0			2			Χ				0		
3										3									
4				Χ	0					4				Χ	0				
5						0				5						0			
6			Χ							6			Χ						
7										7									
8										8									
9	9																		
			Pla	yer	= ' X'	,													
		P	os.g	etR	ow =	= 0													
		Pos	.ge	tCol	umr	n = 0													

void placeMarker(BoardPosition pos, char player)

				Inpu	ıt				I			Exi	ect	ed (Outr	outs			Reason for Test Case
				State										Stat	•				This tests the case where a
	0	1	2	3	4	5	6	7			0	1	2	3	4	5	6	7	player is added at the third
0										0									quadrant's bound.
1										1									
2			Χ				0			2			Χ				0		
3										3									
4				Χ	0					4				Χ	0				
5						0				5						0			
6			Χ							6			Χ						
7										7									
8										8									
9	9																		
			Pla	yer	= 'X'	,													
	Pos	_	_			Rows	5												
		Pos	s.ge	tCol	umr	n = 0													

void placeMarker(BoardPosition pos, char player)

				Inpu	ıt						Exp	oect	ed (Outp	outs			Reason for Test Case
			,	Stat	e:							Ç	Stat	e:				The case tests the situation
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7	where a new player is
0									0									added to the game board.
1									1									
2			Χ				0		2			Χ		Z		0		
3									3									
4				Χ	0				4				Χ	0				
5						0			5						0			
6			Χ						6			Χ						
7									7									
8									8									
9									9									
			os.g	etR	= 'Z' ow = umr													