## CheckerBoard(void) – [Test\_CheckerBoard\_Constructor]

Input: N/A	Output: N/A									
State: N/A	State:									
	pieceCount: x: 12 o: 12 viableDirections: x: [SE, SW] o: [NE, NW] board:									
	х	*	x	*	х	*	х	*		
	*	х	*	х	*	х	*	х		
	х	*	х	*	х	*	х	*		
	*		*		*		*			
		*		*		*		*		
	*	0	*	o	*	o	*	О		
	0	*	0	*	o	*	0	*		
	*	0	*	o	*	o	*	0		

# whatsAtPos(BoardPosition pos) – [test\_whatsAtPos\_MinRowMinCol]

Input: (0,0)	Output: x
State: pieceCount:	State:
x: 12 o: 12	State of the object is unchanged
viableDirections: x: [SE, SW] o: [NE, NW]	

boar	d:									
х	*	х	*	х	*	х	*			
*	х	*	х	*	х	*	х			
х	*	х	*	х	*	х	*			
*		*		*		*				
	*		*		*		*			
*	0	*	0	*	0	*	0			
0	*	0	*	0	*	0	*			
*	0	*	0	*	0	*	О			
/hats	AtPo	os(Bo	ardP	ositic	n po	s) – [	test_v			
Inpu	t: (7,	7)								
State	е:									
State: pieceCount: x: 12 o: 12										
viabl x: [S o: [N	E, S		ns:							

board:

Х

Х

Х

Χ

Χ

Х

\*

Х

Х

Х

\*

Х

Х

X

## whatsAtPos(BoardPosition pos) – [test\_whatsAtPos\_MidRowMidCol]

In	put:	(2.	4

State:

pieceCount:

x: 12 o: 12

viableDirections:

x: [SE, SW] o: [NE, NW]

board:

х	*	х	*	х	*	х	*
*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	* O	*	* O	*	* O	*	* O
*		*		*		*	

Output: x

State:

State of the object is unchanged

# whatsAtPos(BoardPosition pos) – [test\_whatsAtPos\_BlackTile]

Input: (0, 1)							Output: *					
State	<b>e</b> :							State:				
piece x: 12 o: 12	o: 12							State of the object is unchanged				
viableDirections: x: [SE, SW] o: [NE, NW] board:												
х	*	х	*	х	*	х	*					
*	х	*	х	*	х	*	х					
х	*	х	*	х	*	х	*					
*		*		*		*						
	*		*		*		*					
*	0	*	0	*	0	*	0					
o	*	0	*	0	*	0	*					
*	О	*	0	*	0	*	0					

whatsAtPos(BoardPosition pos) - [test\_whatsAtPos\_EmptyTile

Inpu	t: (4,	0)						Output: " " (empty space)
State	е.							State:
	eCou	ınt:						
x: 12 o: 12								State of the object is unchanged
مامان	la Dia	4:						
	iedire SE, SI	ectior W]	15.					
o: [N	IE, N	W]						
boar	d:							
X	*	Х	*	Х	*	Х	*	
*	х	*	х	*	х	*	х	
x	*	х	*	х	*	x	*	
*		*		*		*		
	*		*		*		*	
*	0	*	0	*	0	*	0	
0	*	0	*	0	*	0	*	
*	0	*	0	*	О	*	0	
	D:	<b>(D</b>		.,.				
place	Piece	e(Ros	ardPo	Sition	n pos	, cha	r play	er) – [test_placePiece_EmptyTile_player]

# X]

Input: pos = (3, 3) player = "x"	Output: N/A State:
State: pieceCount: x = 12 o = 12	pieceCount: x = 13 o = 12
viableDirections: x = [SE, SW] o = [NE, NW]	viableDirections: x = [SE, SW] o = [NE, NW]

*	х	*	х	*	Х	*
х	*	x	*	х	*	х
*	х	*	х	*	х	*
	*		*		*	
*		*		*		*
0	*	0	*	0	*	0
*	0	*	0	*	0	*
0	*	0	*	0	*	0
	x * * O *	* X * * X * * * * * * * * * * * * * * *	X * X * X * * * * * * * * * * * * * * *	X	X	X

Х	*	х	*	х	*	х	*
*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*
*		*	х	*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
o	*	0	*	0	*	0	*
*	О	*	0	*	О	*	0

### placePiece(BoardPosition pos, char player) – [test\_placePiece\_EmptyTile\_playerO]

Output: N/A Input: pos = (4,4)player = "o" State: pieceCount: State: x = 12o = 13 pieceCount: x = 12o = 12 viableDIrections: x = [SE, SW]o = [NE, NW] viableDIrections: x = [SE, SW]o = [NE, NW]Χ Χ Χ Χ Χ Χ Χ Χ Χ Χ Х Χ Χ Χ Χ Χ Χ Χ Χ Χ Χ Χ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

*	0	*	0	*	0	*	0		*	0	*	0	*	0	*	0	
									-								

placePiece(BoardPosition pos, char player) – [test\_placePiece\_occupied\_replaceO\_withX]

Input: pos = (5,3) player = "x"

State: pieceCount:

x = 12o = 12

viableDlrections: x = [SE, SW] o = [NE, NW]

х	*	x	*	x	*	x	*
*	х	*	х	*	х	*	х
х	*	Х	*	Х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
О	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: N/A

State:

pieceCount: x = 13

o = 11

viableDIrections:

x = [SE, SW] o = [NE, NW]

*	х	*	х	*	х	*
х	*	х	*	х	*	х
*	х	*	х	*	х	*
	*		*		*	
*		*		*		*
О	*	х	*	0	*	0
*	0	*	0	*	0	*
О	*	0	*	0	*	0
	* * * * * *	X * X * X * X * X * X * X * X * X * X *	X	X	X	X

placePiece(BoardPosition pos, char player) – [test\_placePiece\_occupied\_replaceX\_withO]

Inpu	= (2,0	)) o"						Outp		N/A					
State piece x = 1 o = 1	eCou I2 I2 IeDIre	nt: ectior	ns:					piec x = 7 o = 7 viab x = [ o = 1	eCou 11 13 IeDIr SE,	ectic					
	SE, S NE, 1							х	*	х	*	х	*	х	*
х	*	х	*	х	*	х	*	*	х	*	х	*	х	*	х
*	х	*	х	*	х	*	х	0	*	х	*	х	*	х	*
x	*	х	*	х	*	х	*	*		*		*		*	
*		*		*		*			*		*		*		*
	*		*		*		*	*	0	*	0	*	0	*	0
*	0	*	0	*	0	*	0	0	*	0	*	0	*	0	*
О	*	0	*	0	*	0	*	*	o	*	0	*	o	*	0
*	0	*	0	*	0	*	0								
			•												

# placePiece(BoardPosition pos, char player) – [test\_placePiece\_occupied\_replacex\_withX]

Input:	Output: N/A
pos = (1, 3)	
player = "X"	State:
	pieceCount:
State:	x = 12
pieceCount:	o = 12
x = 12	
o = 12	viableDirections:
	x = [SE, SW]
viableDIrections:	o = [NE, NW]
x = [SE, SW]	
o = [NE, NW]	

*	х	*	х	*	Х	*
х	*	х	*	х	*	х
*	x	*	х	*	х	*
	*		*		*	
*		*		*		*
О	*	0	*	0	*	0
*	0	*	0	*	0	*
О	*	0	*	o	*	o
	* * O	X	x	X	X	X

х	*	х	*	х	*	х	*
*	х	*	Х	*	х	*	х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	О	*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

## getPieceCounts(void) - [test\_getPieceCounts\_x12\_o12]

Input: N/A

State:

pieceCount:

x: 12 o: 12

viableDirections:

x: [SE, SW] o: [NE, NW]

board:

х	*	х	*	х	*	х	*
*	х	*	х	*	х	*	х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
o	*	0	*	0	*	0	*

Output:

x: 12 o: 12

State:

State of the object is unchanged

*   0   *   0   *   0   *   0
-------------------------------

#### getViableDirections(void) - [test\_getViableDirections\_8x8board]

Input: N/A

State:

pieceCount:

x = 12

o = 12

viableDIrections:

x = [SE, SW]

o = [NE, NW]

х	*	х	*	х	*	х	*
*	х	*	Х	*	х	*	Х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
o	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output:

HashMap<Character, ArrayList<DirectionEnum>>

ViableDirections: x = [SE, SW], o =

[NE, NW]

State:

State of the pieceCount is unchanged
State of the board in unchanged
State of the HashMap is unchanged

checkPlayerWin(Character player) - [test\_checkPlayerWin\_no\_opponent\_pieces\_left]

Input: x							Output: True
State:							State: State of the object is unchanged
pieceCou x: 12 o: 0	ınt:						unonangea
viableDire x: [SE, S' o: [NE, N board:	W]	ns:					
x *	х	*	х	*	х	*	
* x	*	х	*	х	*	х	
x *	х	*	х	*	х	*	
*	*		*		*		
*		*		*		*	
*	*		*		*		
*		*		*		*	
*	*		*		*		

# checkPlayerWin(Character player) – [test\_checkPlayerWin\_opponent\_pieces\_exist]

Input: x	Output: False
State: pieceCount: x: 12 o: 3	State: State of the object is unchanged
viableDirections: x: [SE, SW] o: [NE, NW] board:	

X	*	х	*	х	*	х	*
*	х	*	х	*	Х	*	х
х	*	х	*	х	*	х	*
*		*		*		*	
	*		*		*		*
*		*		*		*	
	*		*	О	*		*
*	0	*	0	*		*	

## crownPiece(BoardPosition posOfPlayer) - [test\_crownPiece\_playerx\_on\_opposing\_side]

Output: posOfPlayer = [X] Input: (7,1) State: State: pieceCount: pieceCount: x: 12 x: 12 o: 10 o: 10 viableDirections: viableDirections: x: [SE, SW] x: [SE, SW, NE, NW] o: [NE, NW] o: [NE, NW] board: board: Χ Χ Χ Χ Χ Χ Х Χ Χ Χ Χ Χ Х Χ Х Х Χ Х Х 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0

0

Χ

0

Χ

0

0

0

# crownPiece(BoardPosition posOfPlayer) – [test\_crownPiece\_playerx\_on\_player\_side]

Input	t: (0,C	))						Output: N/A
State piece x: 12 o: 12	eCou	nt:						State: State of the object is unchanged
x: [S	eDire E, SV E, N\	V, NE		/]				
x	*	х	*	х	*	Х	*	
*	х	*	х	*	х	*	х	
х	*	х	*	х	*	х	*	
*		*		*		*		
	*		*		*		*	
*	0	*	0	*	0	*	0	
0	*	0	*	0	*	0	*	
*	0	*	0	*	0	*	0	

## crownPiece(BoardPosition posOfPlayer) - [test\_crownPiece\_playerx\_within\_middle]

Input: (4,0)	Output: N/A
State: pieceCount: x: 12 o: 12	State: State of the object is unchanged
viableDirections: x: [SE, SW, NE, NW] o: [NE, NW]	
board:	

х	*	х	*	х	*	х	*
*	х	*	х	*	х	*	х
	*	х	*	х	*	х	*
*		*		*		*	
х	*		*		*	0	*
*	0	*	0	*	0	*	
0	*	О	*	0	*	0	*
*	0	*	0	*	0	*	o
			-		-		

## movePiece(BoardPosition startingPos, DirectionEnum dir) – [test\_movePiece\_in\_empty\_spot]

Input: ((2,0), SE)

State:

pieceCount:

x: 12 o: 12

viableDirections:

x: [SE, SW] o: [NE, NW]

board:

Doar	<u> </u>						
х	*	x	*	x	*	x	*
*	х	*	х	*	х	*	х
х	*	х	*	х	*	Х	*
*		*		*		*	
	*		*		*		*
*	0	*	0	*	0	*	0
О	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: BoardPosition object = (3,1)

State:

startingPos = null

pieceCount:

x: 12 o: 12

viableDirections:

x: [SE, SW] o: [NE, NW]

board:

_								
	X	*	Х	*	х	*	Х	*
	*	х	*	х	*	х	*	Х
		*	х	*	х	*	х	*
	*	х	*		*		*	
		*		*		*		*
	*	0	*	0	*	0	*	0
	0	*	0	*	0	*	0	*

	*	o	*	0	*	0	*	0	

movePiece(BoardPosition startingPos, DirectionEnum dir) – [test\_movePiece\_in\_occupied\_spot]

Input: ((3,1), SW)

State:

pieceCount:

x: 12

o: 12

viableDirections:

x: [SE, SW]

o: [NE, NW]

board:

<u> </u>							
х	*	Х	*	х	*	Х	*
*	Х	*	Х	*	Х	*	Х
	*	х	*	х	*	Х	*
*	х	*		*		*	
0	*		*		*		*
*		*	0	*	0	*	0
o	*	0	*	0	*	0	*
*	0	*	0	*	0	*	0

Output: startingPos(3,1)

State:

pieceCount:

x: 12

o: 12

viableDirections:

x: [SE, SW]

o: [NE, NW]

board:

<u> </u>							
х	*	х	*	х	*	х	*
*	х	*	х	*	х	*	х
	*	х	*	х	*	х	*
*	х	*		*		*	
0	*		*		*		*
*		*	0	*	0	*	0
0	*	0	*	0	*	0	*
*	o	*	0	*	0	*	0

movePiece(BoardPosition startingPos, DirectionEnum dir) – [test\_movePiece\_SW\_out\_of\_bounds]

Inpu	Input: ((2,0), SW)										Output: startingPos(2,0)							
piece x: 12	State: pieceCount: x: 12 o: 12									State: pieceCount: x: 12 o: 12								
x: [S o: [N	viableDirections: x: [SE, SW] o: [NE, NW] board:									viableDirections: x: [SE, SW] o: [NE, NW] board:								
х	*	х	*	х	*	х	*		х	*	х	*	х	*	х	*		
*	х	*	х	*	х	*	х		*	х	*	х	*	х	*	х		
х	*	х	*	х	*	х	*		х	*	х	*	х	*	х	*		
*		*		*		*			*		*		*		*			
	*		*		*		*			*		*		*		*		
*	0	*	0	*	0	*	0		*	0	*	0	*	0	*	О		
0	*	0	*	0	*	o	*		0	*	o	*	0	*	0	*		
*	0	*	0	*	О	*	О		*	0	*	0	*	0	*	0		

# jumpPiece(BoardPosition startingPos, DirectionEnum dir) – [test\_jumpPiece\_SEjump]

Input:	Output:
startingPos = new BoardPosition(4,4)	x jumps over to (2,6)
dir = DirectionEnum.SE	o is removed
	True
State:	
viableDirections:	State:
x: [SE, SW]	viableDirections:
o: [NE, NW]	x: [SE, SW]
	o: [NE, NW]
x is at (5,5)	
0 is at (6,4)	pieceCount
	x: 1
pieceCount	o: 0
x: 1	
o: 1	

boar	pard:									board:								
	*		*		*		*			*		*		*		*		
*		*		*		*			*		*		*		*			
	*		*		*		*			*		*		*	х	*		
*		*		*	0	*			*		*		*		*			
	*		*	х	*		*			*		*		*		*		
*		*		*		*			*		*		*		*			
	*		*		*		*			*		*		*		*		
*		*		*		*			*		*		*		*			

# jumpPiece(BoardPosition startingPos, DirectionEnum dir) – [test\_jumpPiece\_SWjump]

Input: startingP dir = Dire	x ju o is	Output: x jumps over to (4,0) o is removed True													
State:	4:					04-									
viableDir		S:				Sta		rectio	ne.						
o: [NE, N	-						SE, S		,,,,,						
	-						۷E, ۱								
pieceCou	ınt				nic.										
x: 1 o: 1							pieceCount x: 1								
0. 1							0: 0								
x is at (2,	-														
o is at (3	1)						board:								
board:								*		*		*		*	
*		*		*		*	*		*		*		*		
*	*		*		*			*		*		*		*	
*	х	*		*		*	*		*		*		*		
* O	* 0 * * *									*		*		*	

		*		*		*		*		*		*		*		*		
	*		*		*		*				*		*		*		*	
		*		*		*		*		*		*		*		*		
	*		*		*		*											•
i									•									

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – [test\_jumpPiece\_invalidMove\_noPieceToJump]

#### Input:

startingPos = new BoardPosition(3,3) dir = DirectionEnum.SE

#### State:

viableDirections:

x: [SE, SW]

o: [NE, NW]

### pieceCount

x: 1

o: 1

x is at (3,3)

no piece to jump in SE direction

#### board:

	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*	Х	*		*	
	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	

#### Output:

no move is made error and asked to retry

#### State:

viableDirections:

x: [SE, SW]

o: [NE, NW]

#### pieceCount

x: 1

o: 1

State of the board is unchanged

# scanSurroundingPositions(BoardPosition startingPos) – [test\_scanSurroundingPositions\_validMoves]

Input starti State x at ( o (1,	ingPo e: (2,2)	os = r	new E	3oard	lPosit	tion(3	Output: (0,0) " " (2,0) " " (0,2) " "	
boar	d:							can move NW, SE
	*		*		*		*	State:
*	0	*		*		*		State of the board is unchanged
	*	х	*		*		*	
*		*		*		*		
	*		*		*		*	
*		*		*		*		
	*		*		*		*	
*		*		*		*		

# scanSurroundingPositions(BoardPosition startingPos) – [test\_scanSurroundingPositions\_oSurroundings]

Input: startingPo State: pieceCou x: 3		Board	Posit	tion(5	Output: (3,3) (5,3) (3,5) (5,5) " "			
o: 1						can move NW, SE		
viableDire x: [SE,SW					State:			
o: [NE, N\	-				State of the board is unchanged			
board:								
*	*		*		*			

	*		*		*		*	
Ì		*		*		*		*
ĺ	*		*	х	*	Х	*	
j		*		*	0	*		*
I	*		*	Х	*		*	
Ĭ		*		*		*		*
Ì	*		*		*		*	
•								

scanSurroundingPositions(BoardPosition startingPos) – [test\_scanSurroundingPositions\_noValidMove]

_									
Input: startingPos = new BoardPosition(0,0)  State: pieceCount: x: 1 o: 0  viableDirections x: [SE,SW] o: [NE, NW]								Output: (1,1) " " (0,3) " " (2,0) " "  no valid move directions  State:  State of the board is unchanged	
١,	board	J.							
	х	*		*		*		*	
	*		*		*		*		
		*		*		*		*	
	* * *								
		*		*		*		*	
	*		*		*		*		
ı									1

getDirection(DirectionEnum dir) – [test_get	tDirection_invalidDirection]
Input: dir = DirectionEnum.NE  State: State of the board doesn't affect the function	Output: direction: (1,1) State: State of the object is unchanged

ı

What tests did each team member write? Just tell me the names of the functions (unless for some reason multiple team members wrote functions for the same method. In that case, tell me which tests specifically by giving me the test names)

Laura	CheckerBoard(int) whatsAtPos(BoardPosition) getPieceCounts(void)
Nadia	checkPlayerWin(Character) crownPiece(BoardPosition) movePiece(BoardPosition, DirectionEnum)
Keerthi	jumpPiece(BoardPosition, DirectionEnum) scanSurroundingPositions(BoardPosition) getDirection(DirectionEnum)
Meher	placePiece(BoardPosition, char) getViableDirections(void)