CheckerBoard(void) – [test_CheckerBoard_Constructor]

<table>
<tr>
<td>Input: N/A

State: N/A</td>
<td>Output: N/A

State:

pieceCount:
x: 12
o: 12

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

board:

<table>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td></tr>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td></tr>
<tr><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td></tr>
<tr><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td></tr>
<tr><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td></tr>
</table>
</td>
</tr>
</table>

whatsAtPos(BoardPosition pos) – [test_whatsAtPos_MinRowMinCol]

<table>
<tr>
<td>Input: (0,0)

State:
pieceCount:
x: 12
o: 12

viableDirections:
x: [SE, SW]
o: [NE, NW]</td>
<td>Output: x

State:

State of the object is unchanged</td>
</tr>
</table>

board:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

whatsAtPos(BoardPosition pos) – [test_whatsAtPos_MaxRowMaxCol]

Input: (7, 7)

State:

pieceCount:
x: 12
o: 12

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

board:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*

Output: o

State:

State of the object is unchanged

*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

whatsAtPos(BoardPosition pos) – [test_whatsAtPos_MidRowMidCol]

Input: (2, 4)

State:

pieceCount:
x: 12
o: 12

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

board:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: x

State:

State of the object is unchanged

whatsAtPos(BoardPosition pos) – [test_whatsAtPos_BlackTile]

<table>
<tr>
<td>
Input: (0, 1)

State:

pieceCount:
x: 12
o: 12

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

board:
<table>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td></tr>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td></tr>
<tr><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td></tr>
<tr><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td></tr>
<tr><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td></tr>
</table>
</td>
<td>
Output: *

State:

State of the object is unchanged
</td>
</tr>
</table>

whatsAtPos(BoardPosition pos) – [test_whatsAtPos_EmptyTile

<table>
<tr>
<td>Input: (4, 0)

State:

pieceCount:
x: 12
o: 12

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

board:

<table>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td></tr>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td></tr>
<tr><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td></tr>
<tr><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td></tr>
<tr><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td></tr>
</table>
</td>
<td>Output: “ ”
(empty space)

State:

State of the object is unchanged</td>
</tr>
</table>

placePiece(BoardPosition pos, char player) – [test_placePiece_EmptyTile_playerX]

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

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*	x	*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

placePiece(BoardPosition pos, char player) – [test_placePiece_cornerTile_replaceO_withX]

Input:
pos = (7, 7)
player = “o”

State:

pieceCount:
x = 12
o = 12

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

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*

Output: N/A

State:
pieceCount:
x = 13
o = 11

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

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	x

*	o	*	o	*	o	*	o

placePiece(BoardPosition pos, char player) – [test_placePiece_WhiteTile]

Input:
pos = (3,0)
player = “x”

State:
pieceCount:
x = 12
o = 12

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

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: N/A

State:

pieceCount:
x = 13
o = 12

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

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
x		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

placePiece(BoardPosition pos, char player) – [test_placePiece_middleTile_replaceX_withO]

Input:
pos = (2,4)
player = “o”

State:

pieceCount:
x = 12
o = 12

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

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: N/A

State:

pieceCount:
x = 11
o = 13

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

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	o	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

placePiece(BoardPosition pos, char player) – [test_placePiece_CornerTile_replacex_withX]

Input:
pos = (0, 0)
player = “X”

State:
pieceCount:
x = 12
o = 12

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

Output: N/A

State:
pieceCount:
x = 12
o = 12

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

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

X	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

getPieceCounts(void) – [test_getPieceCounts_x12_o12]

Input: N/A

State:

pieceCount:
x: 12
o: 12

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

board:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*

Output:
x: 12
o: 12

State:

State of the object is unchanged

*	o	*	o	*	o	*	o

getViableDirections(void) – [test_getViableDirections_8x8board]

Input: N/A

State:

pieceCount:
x = 12
o = 12

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

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

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]

<table>
<tr>
<td>Input: x

State:

pieceCount:
x: 12
o: 0

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

board:
<table>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td></tr>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td></tr>
<tr><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td></tr>
<tr><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td></tr>
</table>
</td>
<td>Output: True

State: State of the object is unchanged</td>
</tr>
</table>

checkPlayerWin(Character player) – [test_checkPlayerWin_opponent_pieces_exist]

<table>
<tr>
<td>Input: x

State:

pieceCount:
x: 12
o: 12

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

board:</td>
<td>Output: False

State: State of the object is unchanged</td>
</tr>
</table>

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

crownPiece(BoardPosition posOfPlayer) – [test_crownPiece_playerx_on_opposing_side]

Input: (7,1)

State:
pieceCount:
x: 12
o: 10

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

board:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*		*	x	*	x	*
*		*		*		*	
	*		*	o	*	o	*
*		*	o	*	o	*	o
	*		*	o	*	o	*
*	x	*	o	*	o	*	o

Output: posOfPlayer = [X]

State:
pieceCount:
x: 12
o: 10

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

board:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*		*	x	*	x	*
*		*		*		*	
	*		*	o	*	o	*
*		*	o	*	o	*	o
	*		*	o	*	o	*
*	X	*	o	*	o	*	o

crownPiece(BoardPosition posOfPlayer) – [test_crownPiece_playerx_on_player_side]

<table>
<tr>
<td>Input: (0,0)

State:
pieceCount:
x: 12
o: 12

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

board:
<table>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td></tr>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td></tr>
<tr><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td></tr>
<tr><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td></tr>
<tr><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td></tr>
</table>
</td>
<td>Output: N/A

State:
State of the object is unchanged</td>
</tr>
</table>

crownPiece(BoardPosition posOfPlayer) – [test_crownPiece_playerx_already_crowned]

<table>
<tr>
<td>Input: (4,0)

State:
pieceCount:
x: 12
o: 12

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

board:</td>
<td>Output: N/A

State: State of the object is unchanged</td>
</tr>
</table>

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
	*	x	*	x	*	x	*
*		*		*		*	
X	*		*		*	o	*
*	o	*	o	*	o	*	
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	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:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
x	*	x	*	x	*	x	*
*		*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: BoardPosition object = (3,1)

State:
startingPos = null

pieceCount:
x: 12
o: 12

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

board:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
	*	x	*	x	*	x	*
*	x	*		*		*	
	*		*		*		*
*	o	*	o	*	o	*	o
o	*	o	*	o	*	o	*

	* o * o * o * o

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:

x	*	x	*	x	*	x	*
*	x	*	x	*	x	*	x
	*	x	*	x	*	x	*
*	x	*		*		*	
o	*		*		*		*
*		*	o	*	o	*	o
o	*	o	*	o	*	o	*
*	o	*	o	*	o	*	o

Output: N/A

State:
State of the object is unchanged

movePiece(BoardPosition startingPos, DirectionEnum dir) – [test_movePiece_SW_out_of_bounds]

<table>
<tr>
<td>Input: ((2,0), SW)

State:
pieceCount:
x: 12
o: 12

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

board:
<table>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td></tr>
<tr><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td><td>x</td><td>*</td></tr>
<tr><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td></tr>
<tr><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td><td></td><td>*</td></tr>
<tr><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td></tr>
<tr><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td></tr>
<tr><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td><td>*</td><td>o</td></tr>
</table>
</td>
<td>Output: N/A

State:
State of the object is unchanged</td>
</tr>
</table>

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – [test_jumpPiece_SEjump]

<table>
<tr>
<td>Input:
startingPos = new BoardPosition(4,4)
dir = DirectionEnum.SE

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

x is at (5,5)
0 is at (6,4)

pieceCount
x: 1
o: 1</td>
<td>Output:
x jumps over to (2,6)
o is removed
True

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

pieceCount
x: 1
o: 0</td>
</tr>
</table>

board:

	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*	o	*	
	*		*	x	*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	

board:

	*		*		*		*
*		*		*		*	
	*		*		*	x	*
*		*		*		*	
	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	

jumpPiece(BoardPosition startingPos, DirectionEnum dir) – [test_jumpPiece_SWjump]

Input:
startingPos = new BoardPosition(2,2)
dir = DirectionEnum.SW

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

pieceCount
x: 1
o: 1

x is at (2,2)
o is at (3,1)

board:

	*		*		*		*
*		*		*		*	
	*	x	*		*		*
*	o	*		*		*	

Output:
x jumps over to (4,0)
o is removed
True

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

pieceCount
x: 1
o: 0

board:

	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	
x	*		*		*		*

	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	

*		*		*		*	
	*		*		*		*
*		*		*		*	

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:

	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*	x	*		*	
	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	

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:
startingPos = new BoardPosition(3,3)

State:
x at (2,2)
o (1,1)

board:

	*		*		*		*
*	o	*		*		*	
	*	x	*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	

Output:
(0,0) “ “
(2,0) “ “
(0,2) “ “

can move NW, SE

State:

State of the board is unchanged

scanSurroundingPositions(BoardPosition startingPos) – [test_scanSurroundingPositions_oSurroundings]

Input:
startingPos = new BoardPosition(4,4)

State:
pieceCount:
x: 3
o: 1

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

board:

	*		*		*		*

Output:
(3,3)
(5,3)
(3,5)
(5,5) “ “

can move NW, SE

State:

State of the board is unchanged

*		*		*		*	
	*		*		*		*
*		*	x	*	x	*	
	*		*	o	*		*
*		*	x	*		*	
	*		*		*		*
*		*		*		*	

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]

board:

x	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	
	*		*		*		*
*		*		*		*	

Output:
(1,1) “ “
(0,3) “ “
(2,0) “ “

no valid move directions

State:

State of the board is unchanged

getDirection(DirectionEnum dir) – [test_getDirection_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

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)