Jeffrey Black
CPSC 2150

How to Play

1. Open a linux terminal and navigate to the folder that contains the folder “cpsc2150” and the file “makefile.”
2. Type “make”
    a. This will compile the java source files, creating .class versions of all of the source files.
3. Type “make run”
    a. This will run the tic-tac-toe program.
    b. Can be ran multiple times.
4. Type “make clean”
    a. This will remove the .class versions of the source files.

Jeffrey Black
CPSC 2150

Requirements Analysis

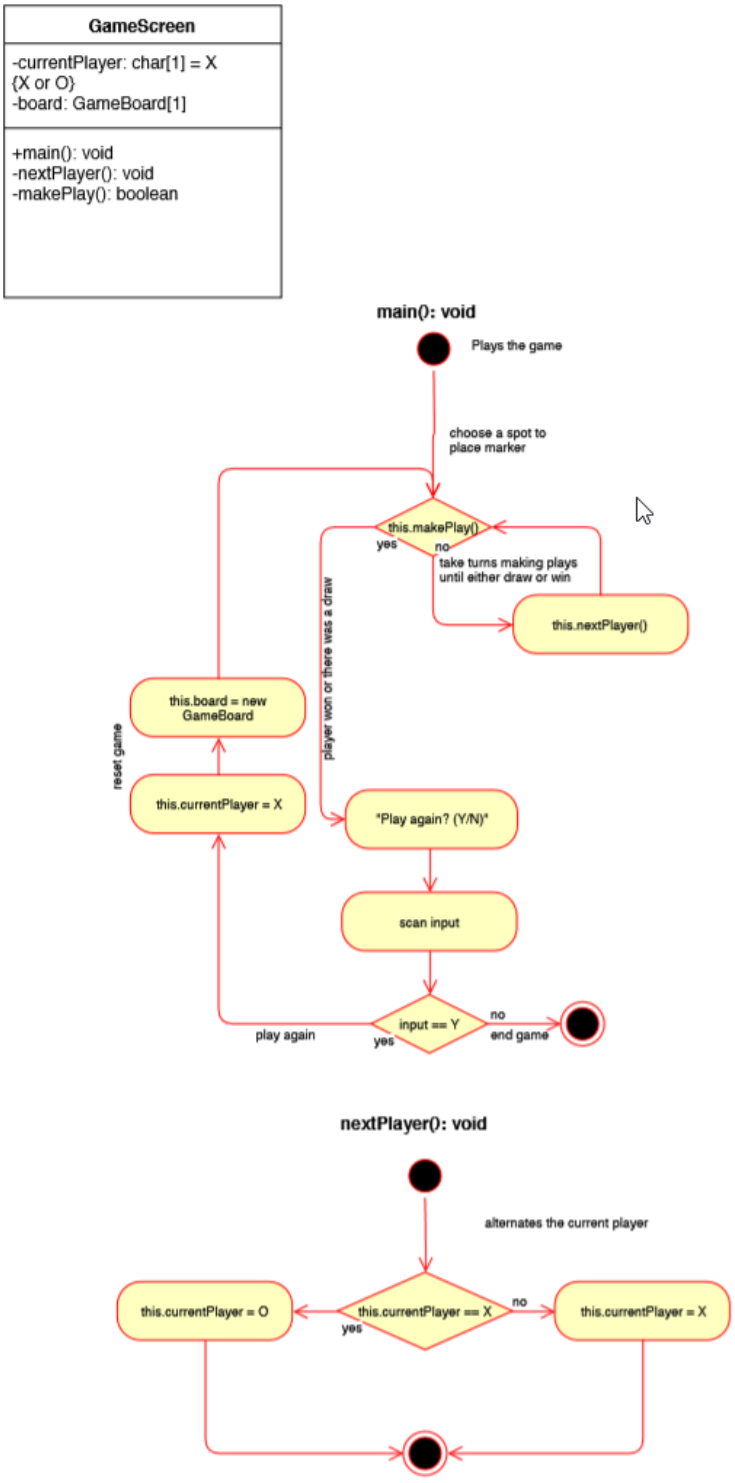
User Stories

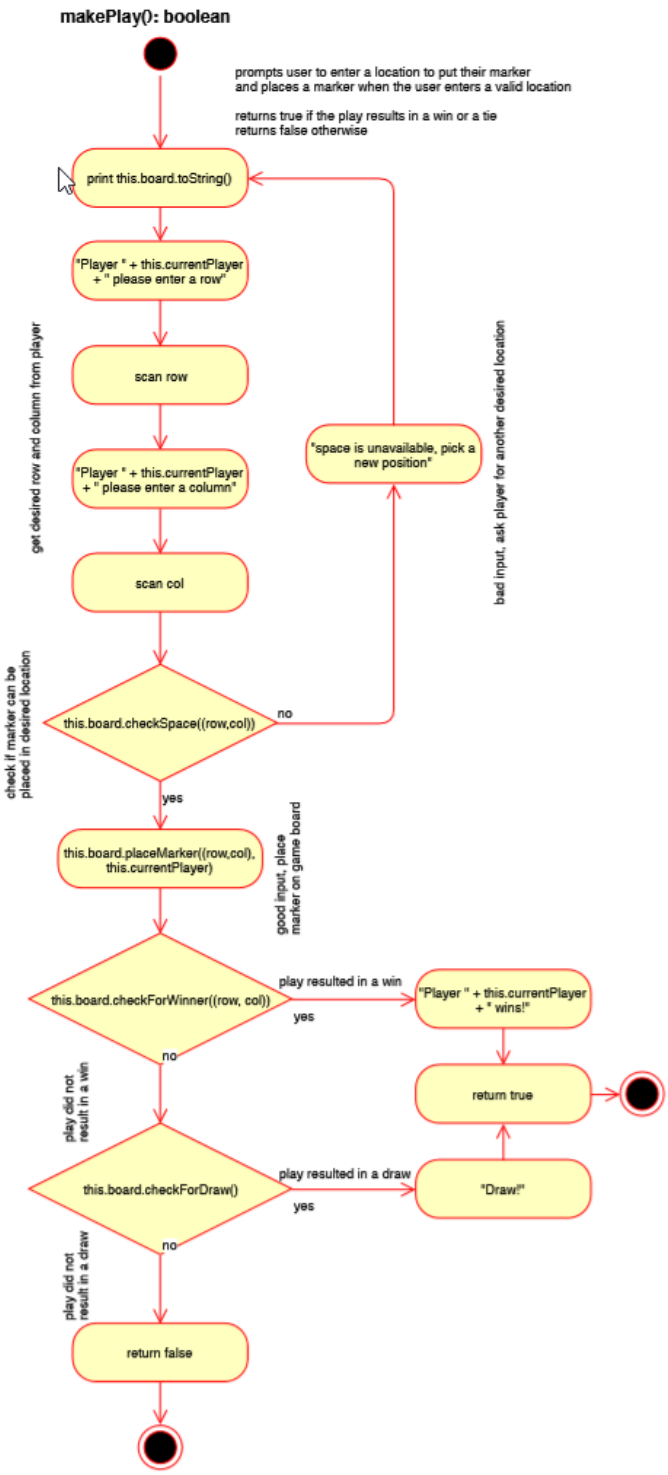
- As a player, I want to be able to place an X or O, so I can play Tic-Tac-Toe
- As a player, I want to be able to see the tic-tac-toe grid, so I can view the locations of the X and O's
- As a player, I want to know when one of the players has won, so I don't have to check after each turn
- As a player, I want to know if there has been a tie, so I don't have to check if there has been a tie
- As a player, I want to know whose turn it is, so I know who has to place an X or O
- As a player, I want to know if I placed my marker on a spot that was already claimed, so that both players don't place their marker on the same spot.
- As a player, I want to know if I placed my marker outside of the grid, so I don't place my marker outside of the playable grid
- As a player, I want to be able to play again after the game ends, so I can play more games without starting the program again

Non-Functional Requirements

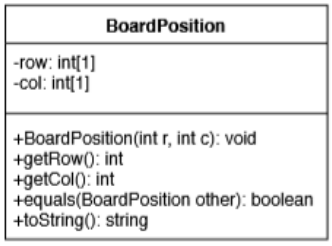
- Must have a grid
- Grid size must be 8 x 8
- 5 in a row constitutes a win
- System must be coded in Java
- System must be able to run on Unix

Jeffrey Black
CPSC 2150

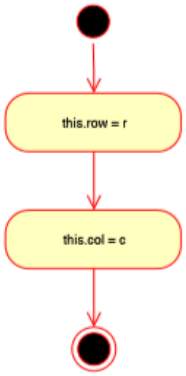



makePlay(): boolean
prompts user to enter a location to put their marker
and places a marker when the user enters a valid location
returns true if the play results in a win or a tie
returns false otherwise
print this.board.toString()
"Player " + this.currentPlayer
+ " please enter a row"
get desired row and column from player
scan row
"Player " + this.currentPlayer
+ " please enter a column"
scan col
"space is unavailable, pick a
new position"
bad input, ask player for another desired location
check if marker can be
placed in desired location
this.board.checkSpace((row,col))
no
yes
this.board.placeMarker((row,col),
this.currentPlayer)
good input, place
marker on game board
this.board.checkForWinner((row, col))
play resulted in a win
yes
"Player " + this.currentPlayer
+ " wins!"
no
return true
play did not
result in a win
this.board.checkForDraw()
play resulted in a draw
yes
"Draw!"
no
play did not
result in a draw
return false

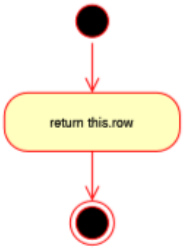
Jeffrey Black
CPSC 2150



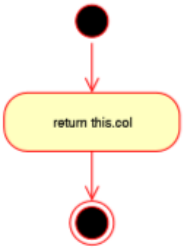
BoardPosition(int r, int c): void



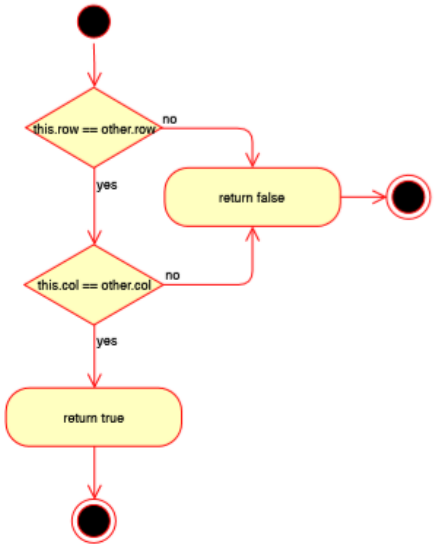
getRow(): int



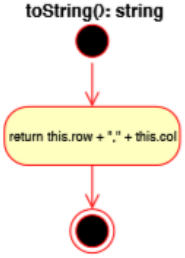
getCol(): int



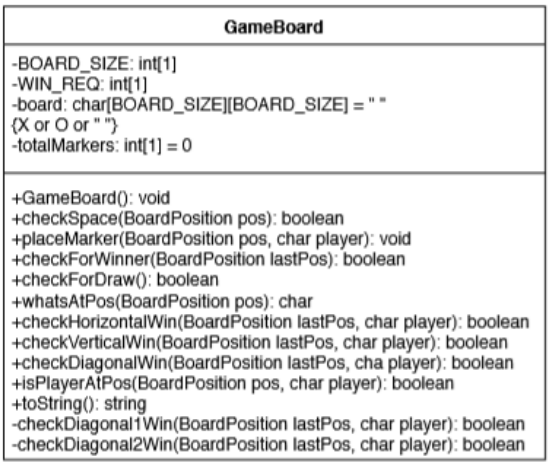
equals(BoardPosition other): boolean

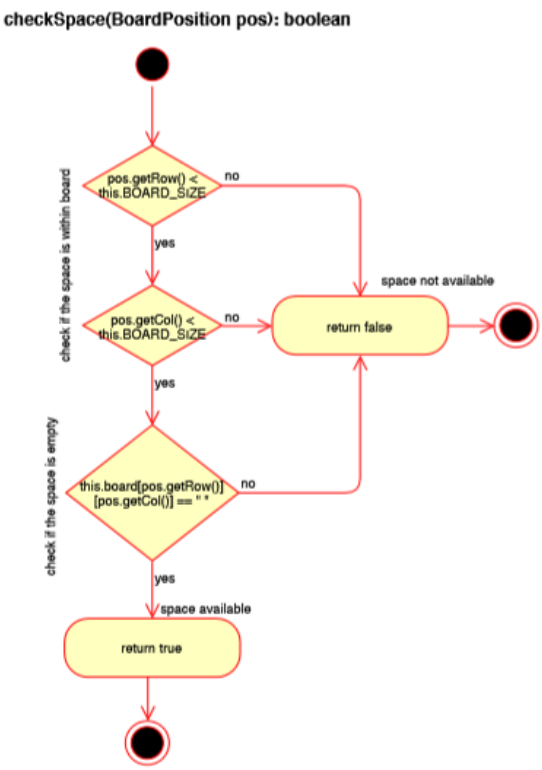


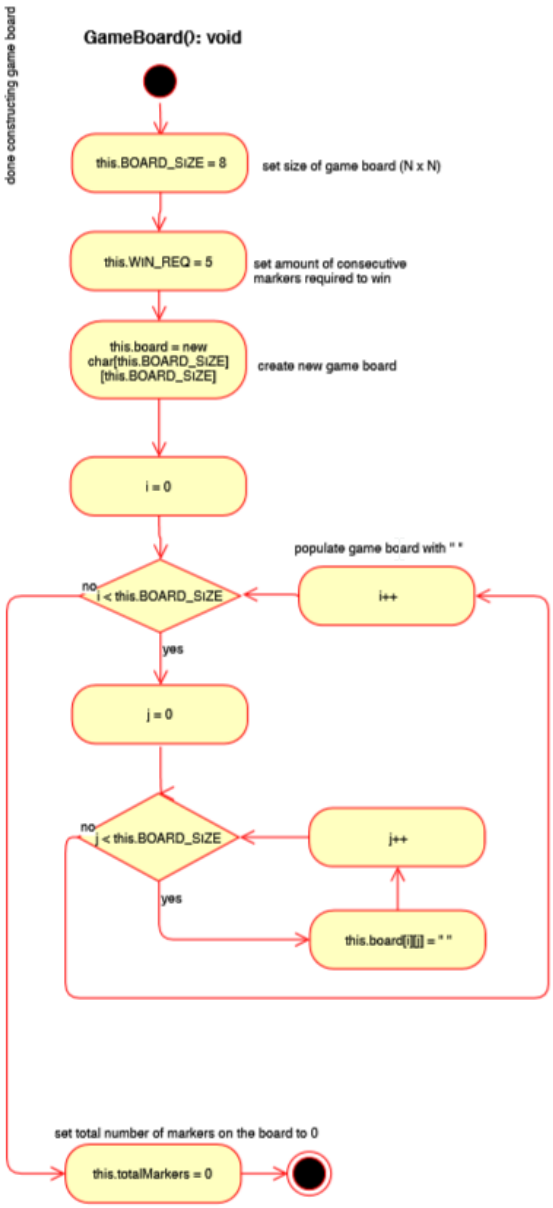
toString(): string



Jeffrey Black
CPSC 2150

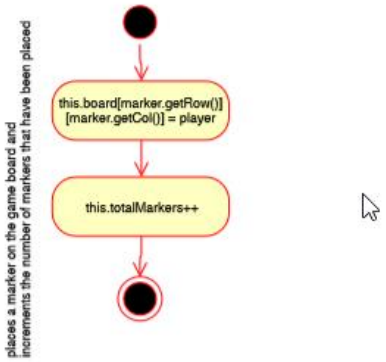




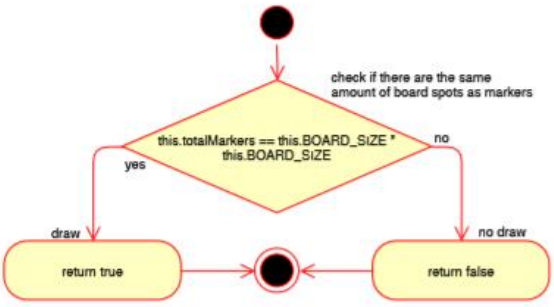


Jeffrey Black
CPSC 2150

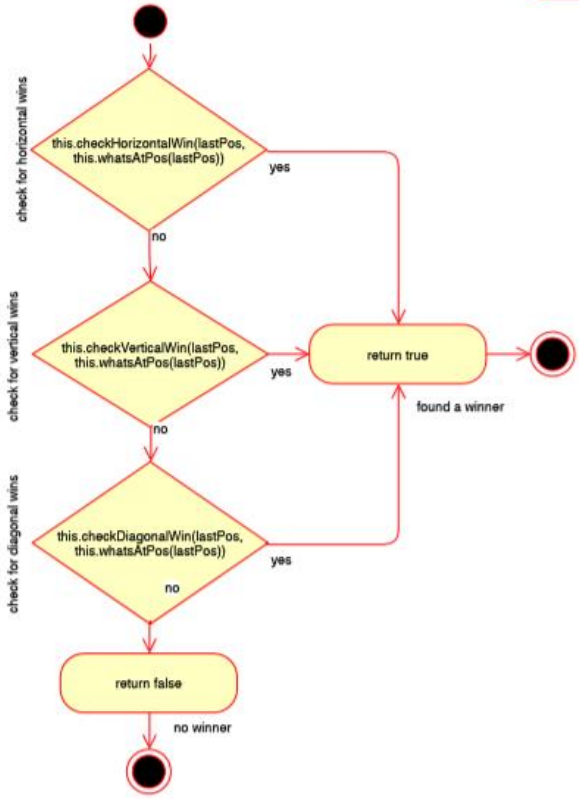
placeMarker(BoardPosition marker, char player): void



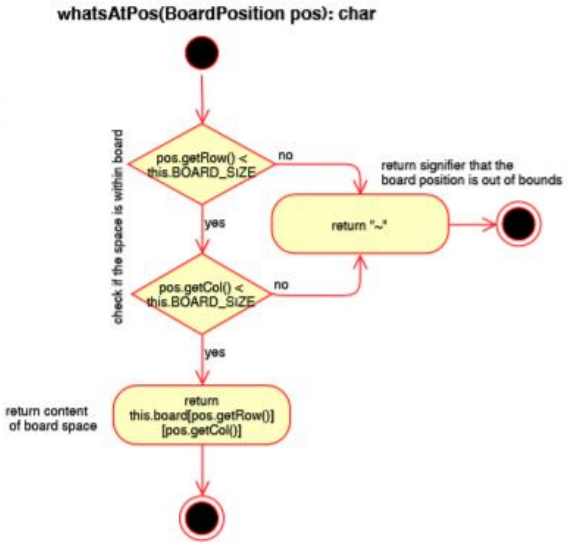
checkForDraw(): boolean

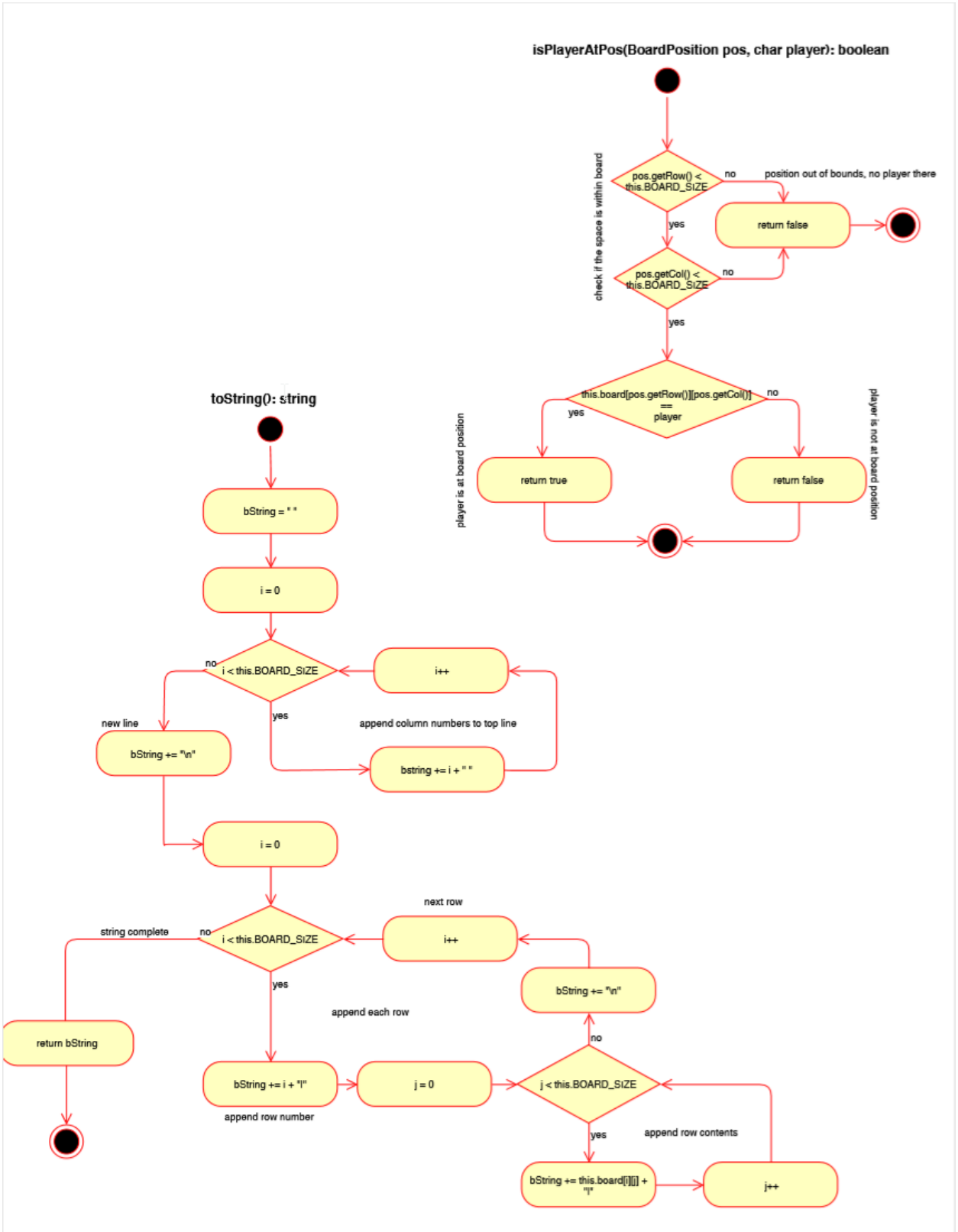


checkForWinner(BoardPosition lastPos): boolean

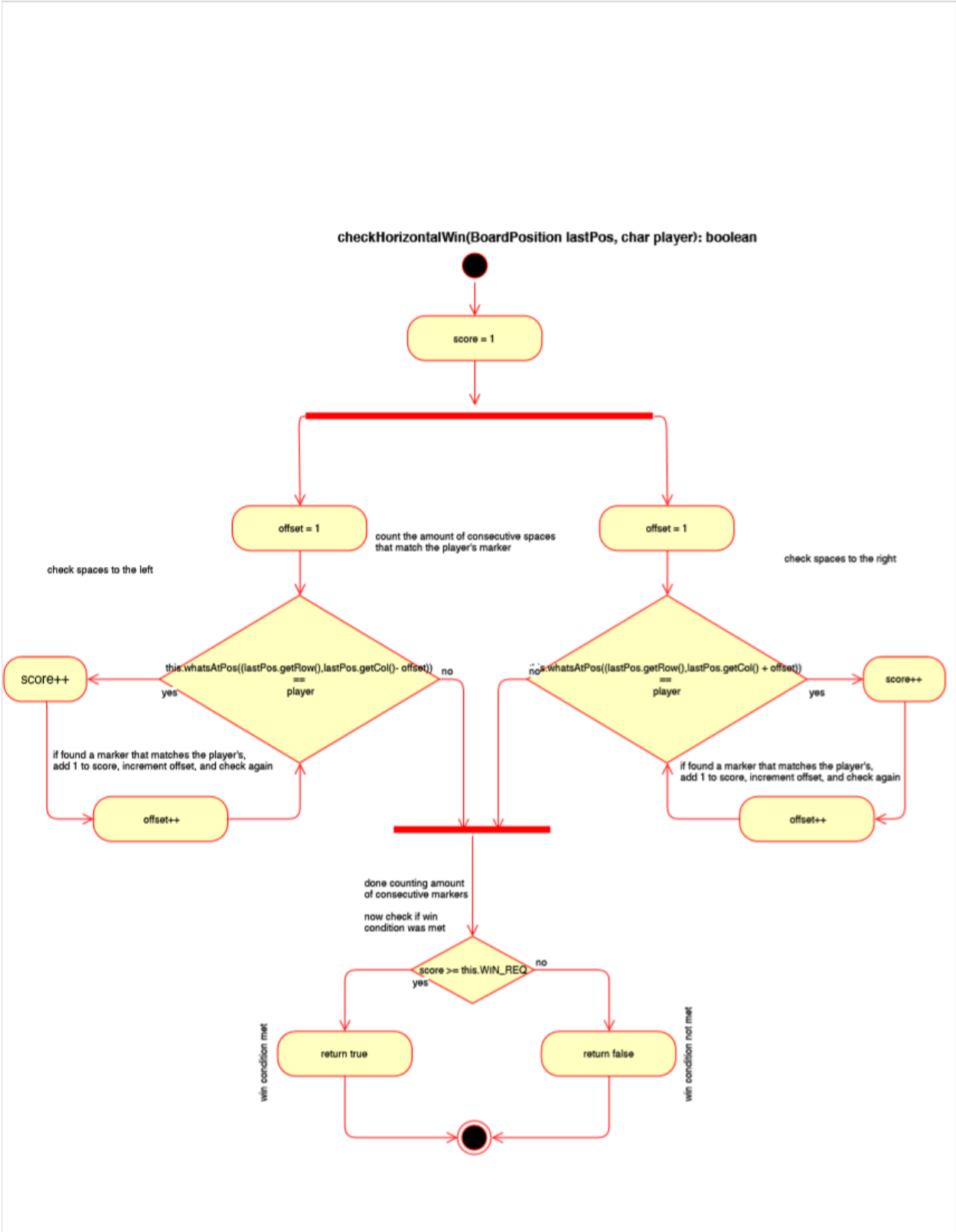


whatsAtPos(BoardPosition pos): char

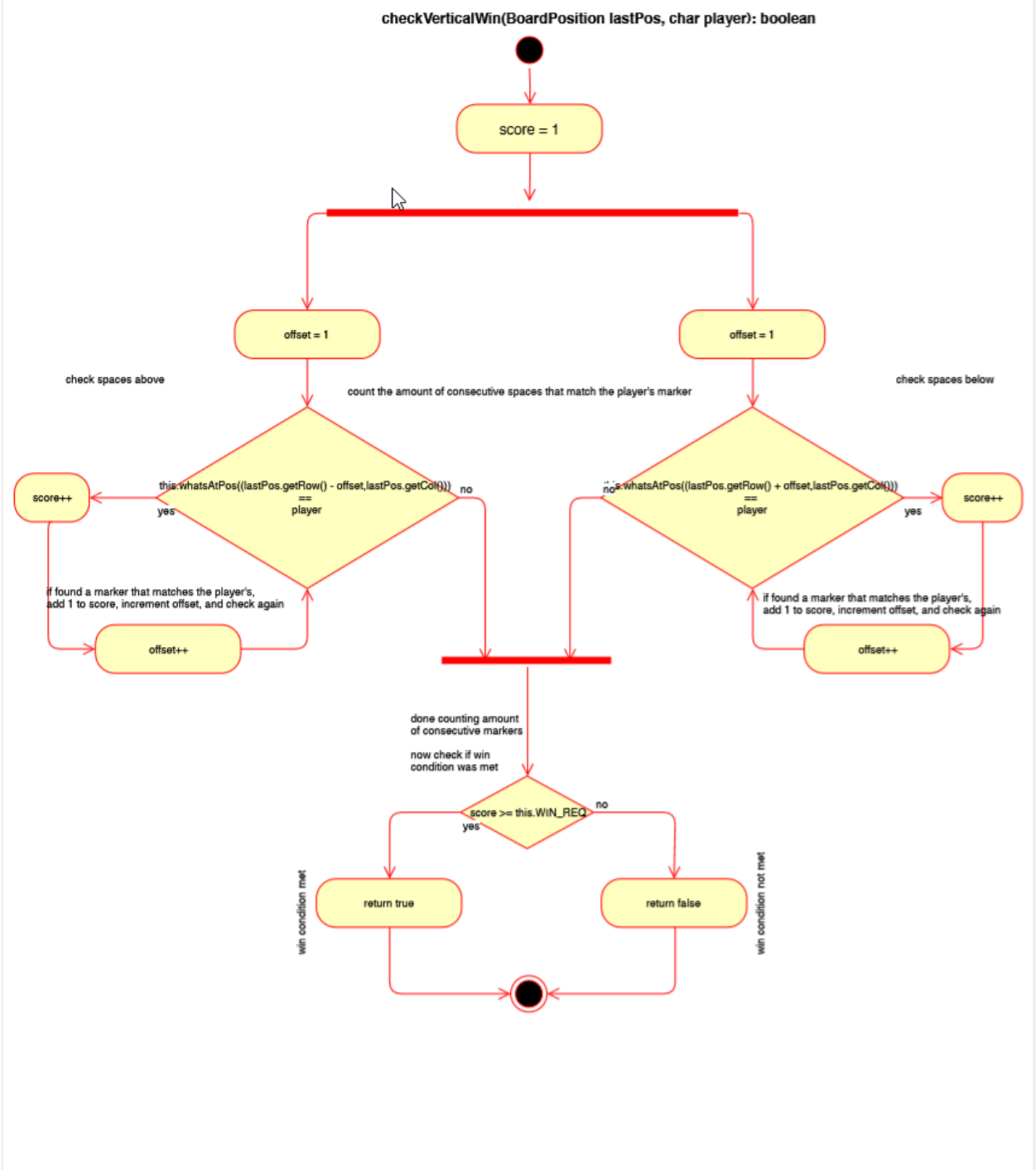


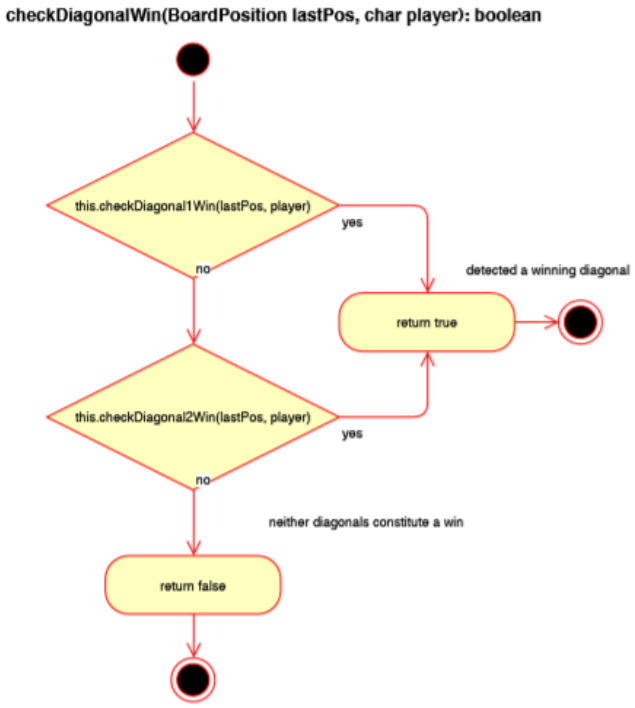

isPlayerAtPos(BoardPosition pos, char player): boolean
check if the space is within board
pos.getRow() < this.BOARD_SIZE
no
position out of bounds, no player there
return false
yes
pos.getCol() < this.BOARD_SIZE
no
yes
this.board[pos.getRow()][pos.getCol()] == player
no
yes
player is at board position
player is not at board position
return true
return false
toString(): string
bString = " "
i = 0
i < this.BOARD_SIZE
no
i++
yes
append column numbers to top line
new line
bString += "\n"
bstring += i + " "
i = 0
next row
string complete
i < this.BOARD_SIZE
no
i++
yes
bString += "\n"
append each row
return bString
bString += i + "|"
append row number
j = 0
j < this.BOARD_SIZE
no
yes
append row contents
bString += this.board[i][j] + "|"
j++

Jeffrey Black
CPSC 2150

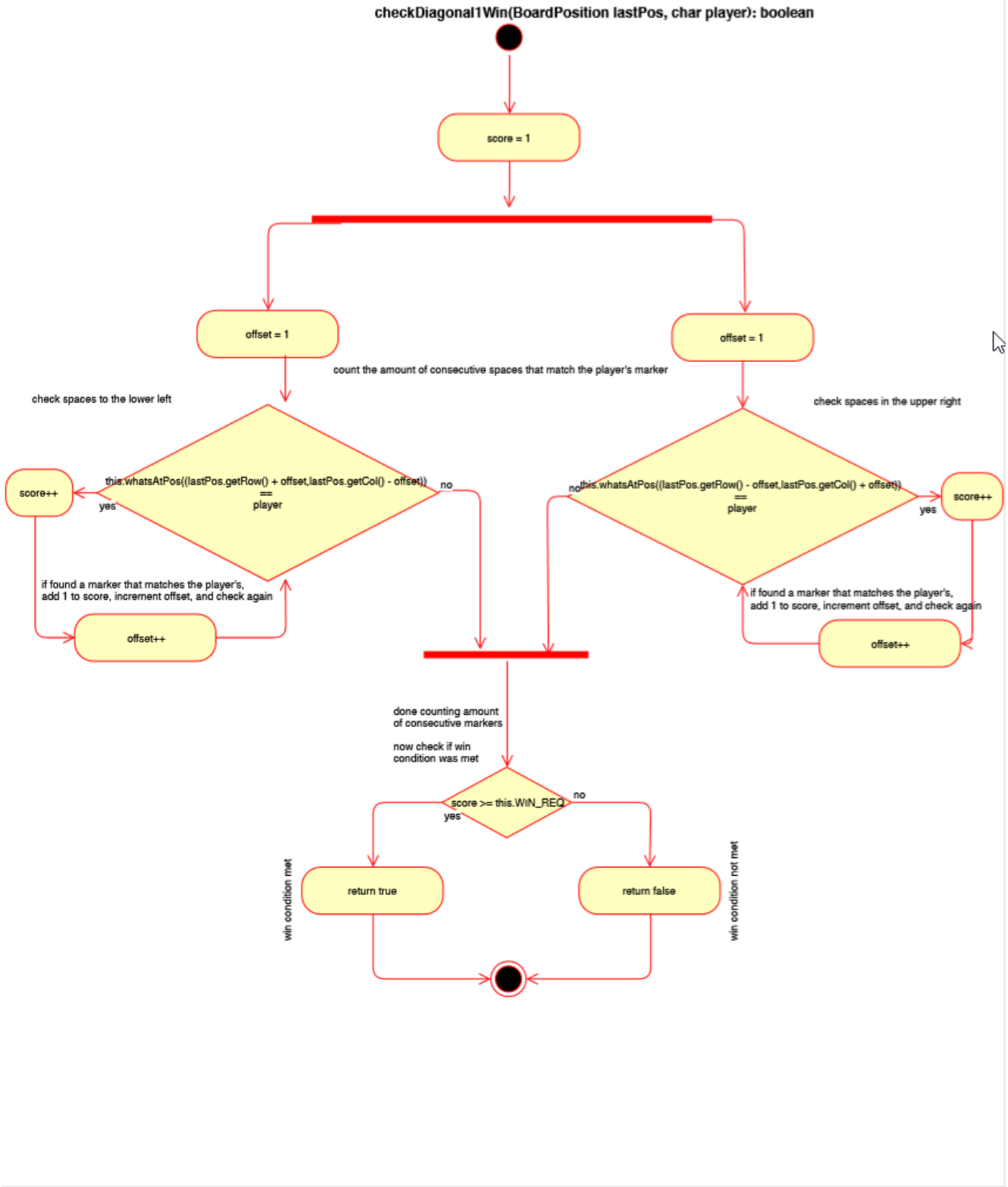


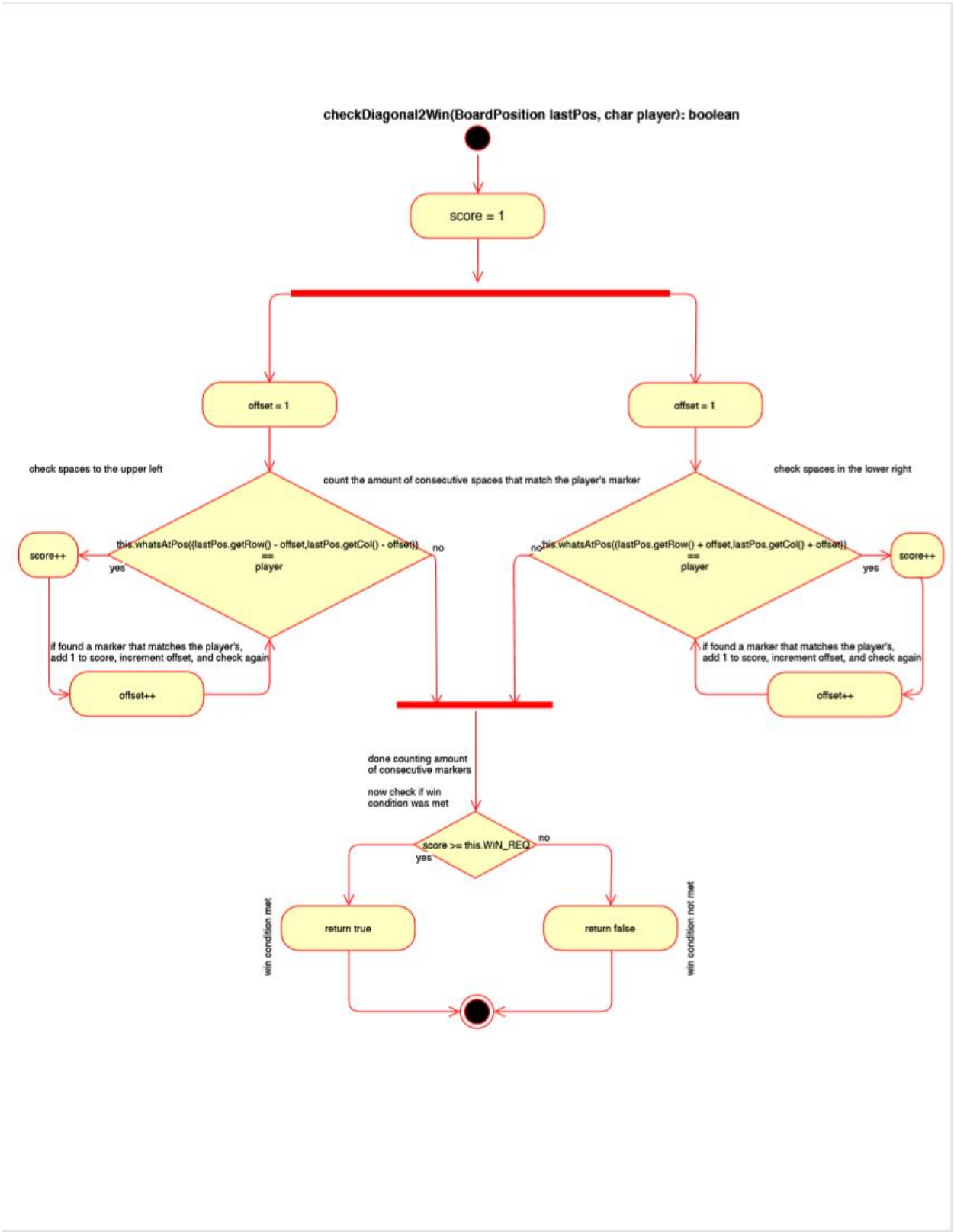
Jeffrey Black
CPSC 2150




checkDiagonalWin(BoardPosition lastPos, char player): boolean
this.checkDiagonal1Win(lastPos, player)
yes
no
detected a winning diagonal
return true
this.checkDiagonal2Win(lastPos, player)
yes
no
neither diagonals constitute a win
return false

Jeffrey Black
CPSC 2150




checkDiagonal2Win(BoardPosition lastPos, char player): boolean
score = 1
offset = 1
offset = 1
check spaces to the upper left
count the amount of consecutive spaces that match the player's marker
check spaces in the lower right
this.whatsAtPos((lastPos.getRow() - offset,lastPos.getCol() - offset)) == player
yes
no
score++
this.whatsAtPos((lastPos.getRow() + offset,lastPos.getCol() + offset)) == player
no
yes
score++
if found a marker that matches the player's, add 1 to score, increment offset, and check again
offset++
if found a marker that matches the player's, add 1 to score, increment offset, and check again
offset++
done counting amount of consecutive markers
now check if win condition was met
score >= this.WIN_REQ
yes
no
win condition met
return true
return false
win condition not met