CPSC 2150 Project Report

Jenna Adams

Requirements Analysis

Functional Requirements:

1. As a player I can drop my x or o into my specified column so that I can take my turn.
2. As a player I need to know when I win so that we don't continue to play a game that is over.
3. As a player I need to know if a column is full so that I know not to drop an x or o in that column.
4. As a player I need to be able to see the game board so that I can strategize my next move.
5. As a player I need to know if the game is a draw so that I don't continue to play a game that is over.
6. As a player I need to be able to see my token or name so that I know if it is my turn.
7. As a player I can see an updated game board after each turn so that I know where my opponent placed their token.
8. As a player I can indicate that I want to play again after the end of the game so that I can play again.
9. As a player I need to know if a tile is already occupied so that I can place my token in a valid tile.
10. As a player I need to know when the game is over so that I can leave the program.
11. As a player, I can pick again if I pick an unavailable column, so I don't lose my turn.
12. As a player, I can pick again if I pick a column that does not exist, so I don't lose my turn.
13. As a player, I can make a move after my opponent does (assuming they don't win), so I can always have my turn.

Non-Functional Requirements

1. The program must be written in Java.
2. The program must run on SoC machines.
3. The game board must render to the screen within 10 seconds of loading the program.
4. The game must be able to run at any hour of the day.
5. The program must not keep previous game data.
6. The board size is six rows by nine columns.
7. The board starts in the bottom left at position 0,0.

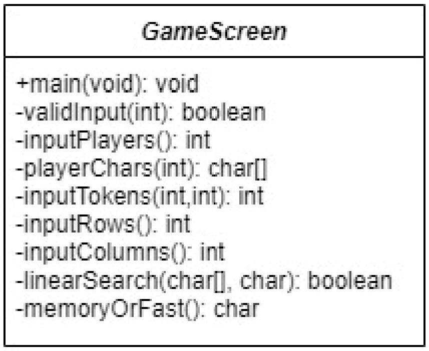
Deployment Instructions

Details in Projects 2-5.

System Design

Class 1: (GameScreen.java)

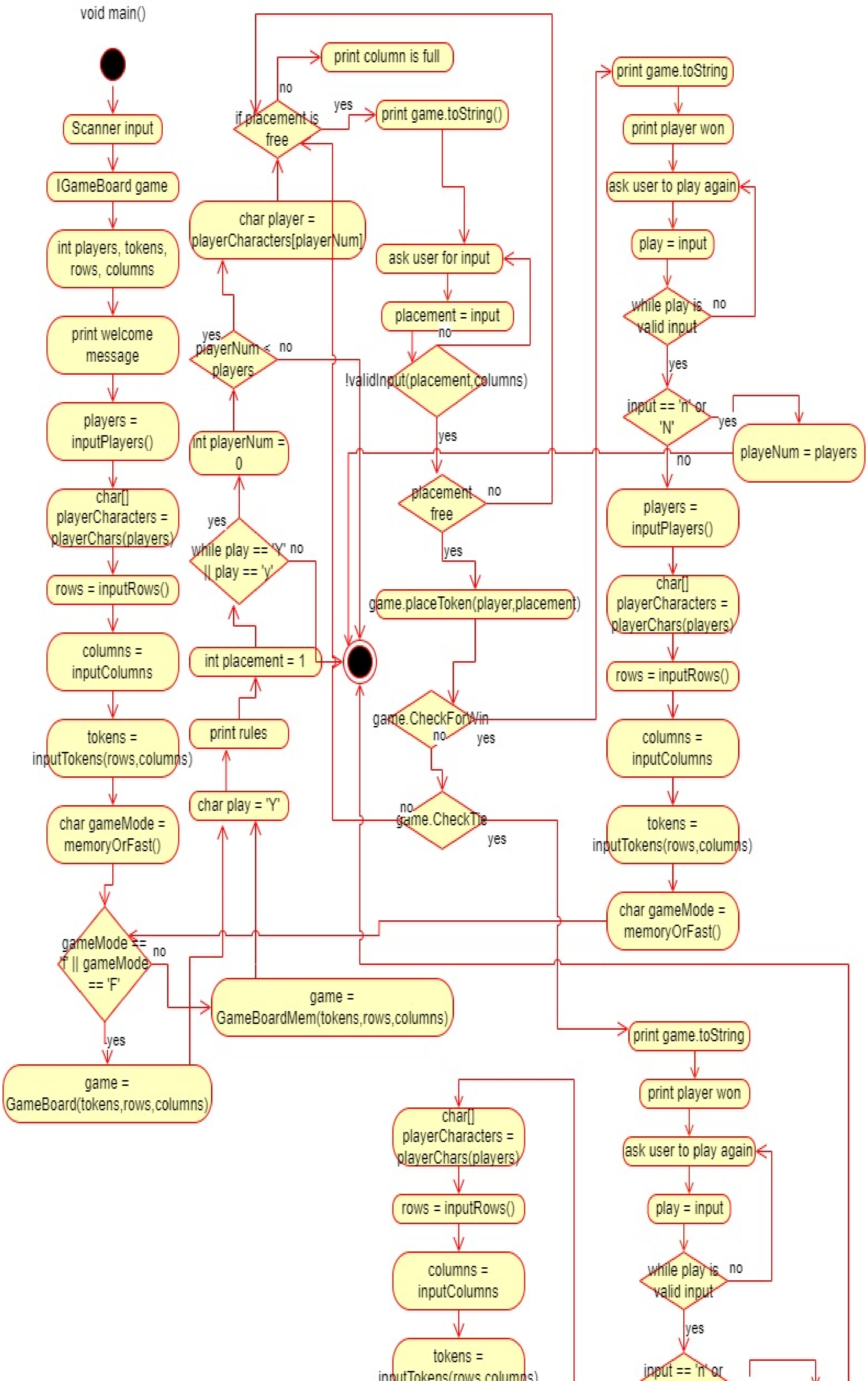
Class diagram



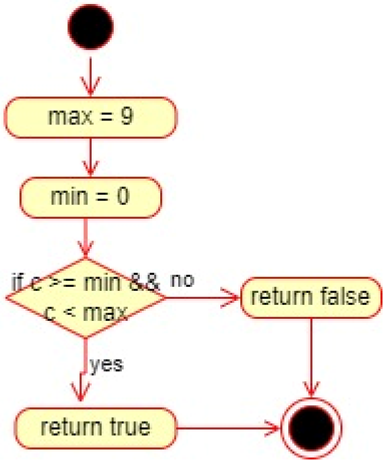
GameScreen

+main(void): void
-validInput(int): boolean
-inputPlayers(): int
-playerChars(int): char[]
-inputTokens(int,int): int
-inputRows(): int
-inputColumns(): int
-linearSearch(char[], char): boolean
-memoryOrFast(): char

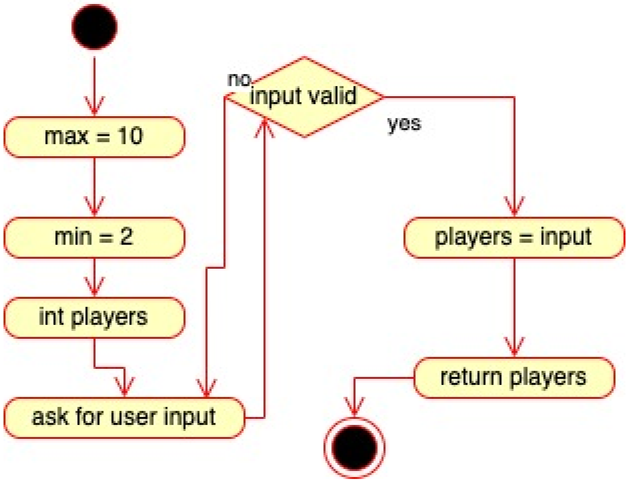
Activity diagrams


void main()
Scanner input
IGameBoard game
int players, tokens, rows, columns
print welcome message
players = inputPlayers()
char[] playerCharacters = playerChars(players)
rows = inputRows()
columns = inputColumns
tokens = inputTokens(rows,columns)
char gameMode = memoryOrFast()
gameMode == 'f' || gameMode == 'F'
no
yes
game = GameBoard(tokens,rows,columns)
game = GameBoardMem(tokens,rows,columns)
char play = 'Y'
print rules
int placement = 1
while play == 'Y' || play == 'y'
yes
no
int playerNum = 0
playerNum < players
yes
no
char player = playerCharacters[playerNum]
if placement is free
no
print column is full
yes
print game.toString()
ask user for input
placement = input
no
!validInput(placement,columns)
yes
placement free
no
yes
game.placeToken(player,placement)
game.CheckForWin
no
yes
no
game.CheckTie
yes
print game.toString
print player won
ask user to play again
play = input
while play is valid input
no
yes
input == 'n' or 'N'
yes
playeNum = players
no
players = inputPlayers()
char[] playerCharacters = playerChars(players)
rows = inputRows()
columns = inputColumns
tokens = inputTokens(rows,columns)
char gameMode = memoryOrFast()
print game.toString
print player won
ask user to play again
play = input
while play is valid input
no
yes
input == 'n' or
char[] playerCharacters = playerChars(players)
rows = inputRows()
columns = inputColumns
tokens = inputTokens(rows,columns)

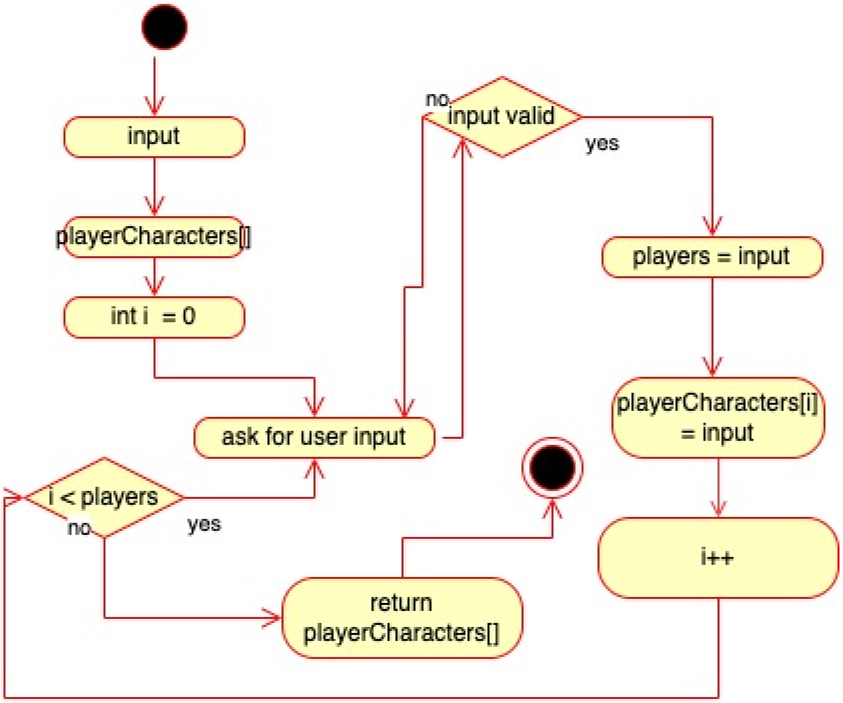
private static boolean vaildInput(int c)



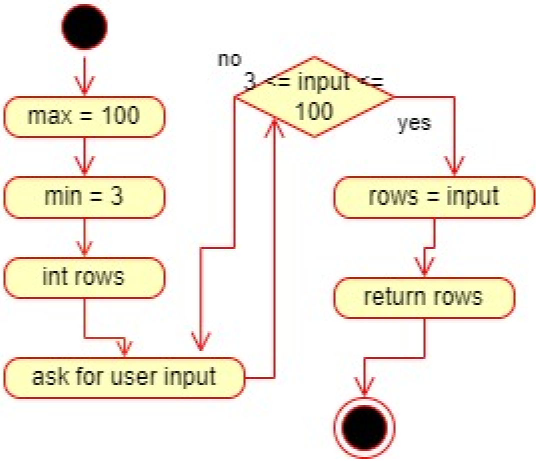
int InputPlayers()



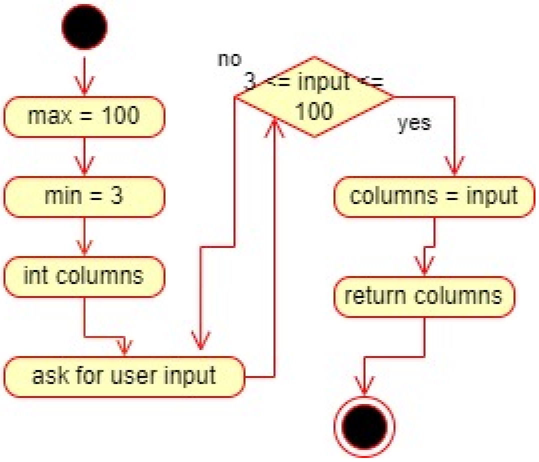
playerChars(int players)



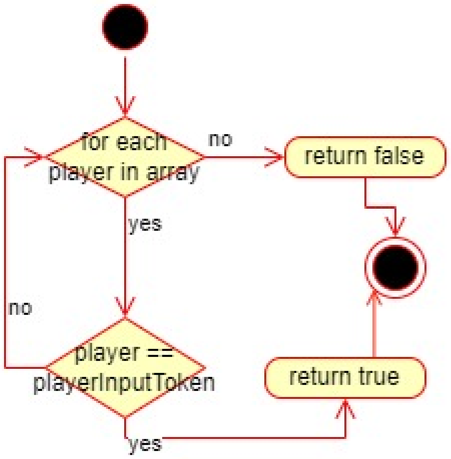
int inputRows()



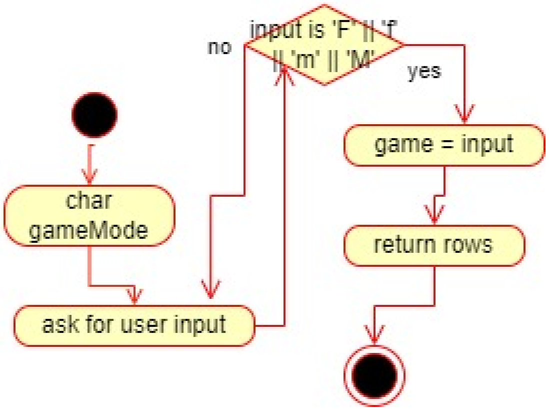
int inputColumns()



boolean linearSearch(char[] array, char playerInputToken)

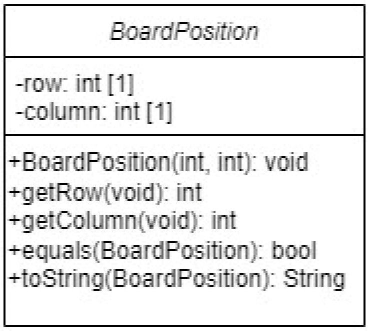


char memoryOrFast()



Class 2: (BoardPosition.java)

Class diagram



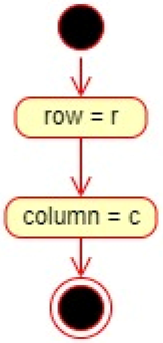
BoardPosition

-row: int [1]
-column: int [1]

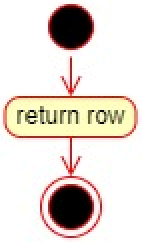
+BoardPosition(int, int): void
+getRow(void): int
+getColumn(void): int
+equals(BoardPosition): bool
+toString(BoardPosition): String

Activity diagrams

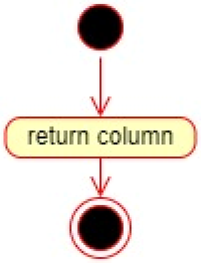
BoardPosition(r, c)



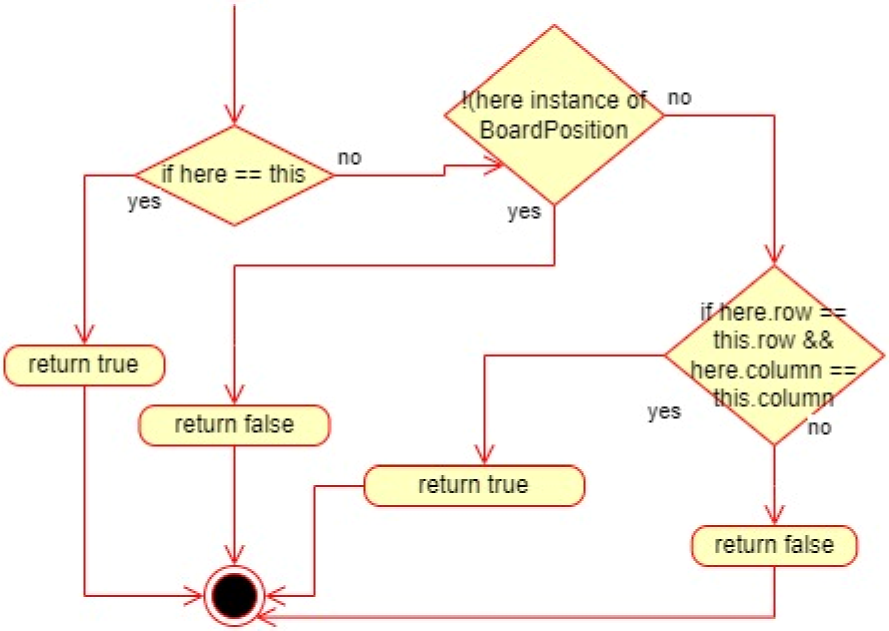
getRow()



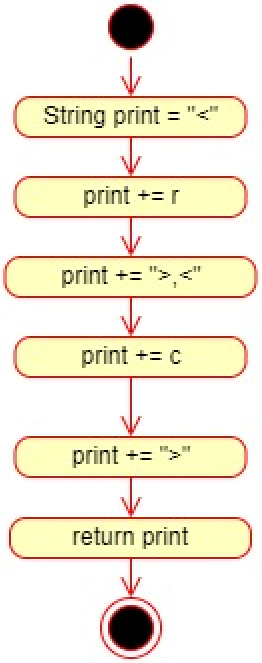
getColumn()



equals(BoardPosition here)

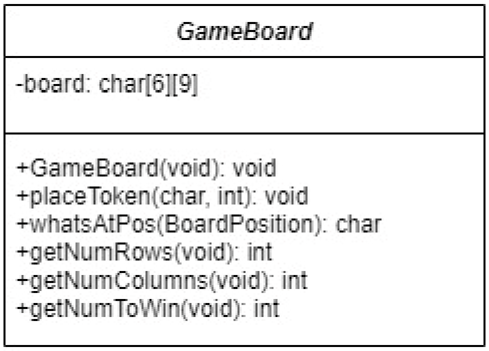

if here == this
yes
no
return true
!(here instance of BoardPosition
yes
no
return false
if here.row == this.row && here.column == this.column
yes
no
return true
return false

toString()



Class 3: (GameBoard.java)

Class diagram



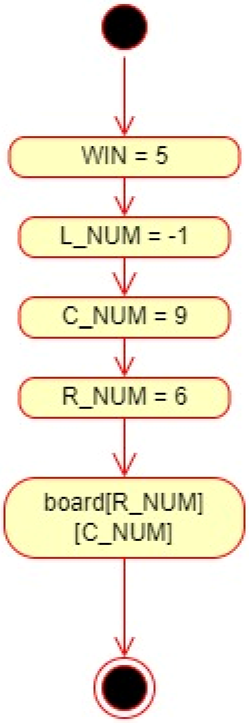
GameBoard

-board: char[6][9]

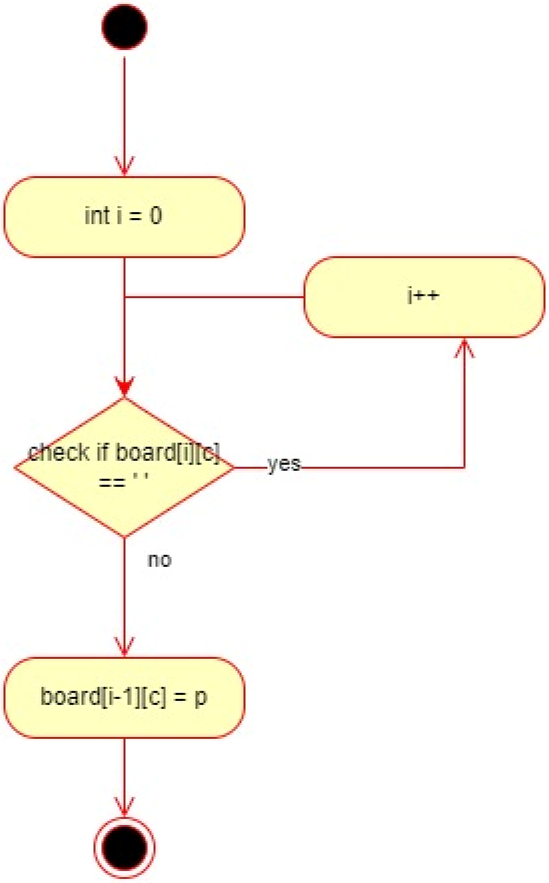
+GameBoard(void): void
+placeToken(char, int): void
+whatsAtPos(BoardPosition): char
+getNumRows(void): int
+getNumColumns(void): int
+getNumToWin(void): int

Activity diagrams

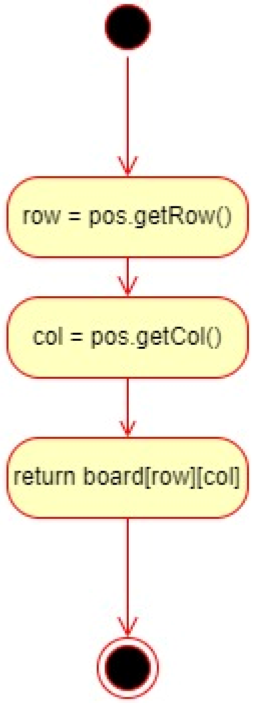
public void GameBoard(void)



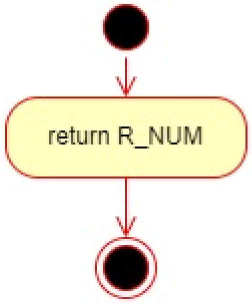
public void placeToken(char p, int c)



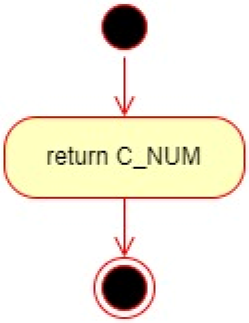
public char whatsAtPos(BoardPosition pos)



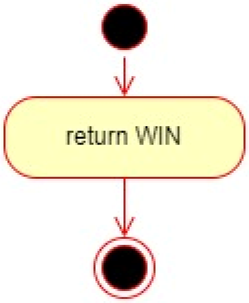
getNumRows()



getNumCols()

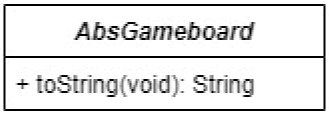


getNumToWin()



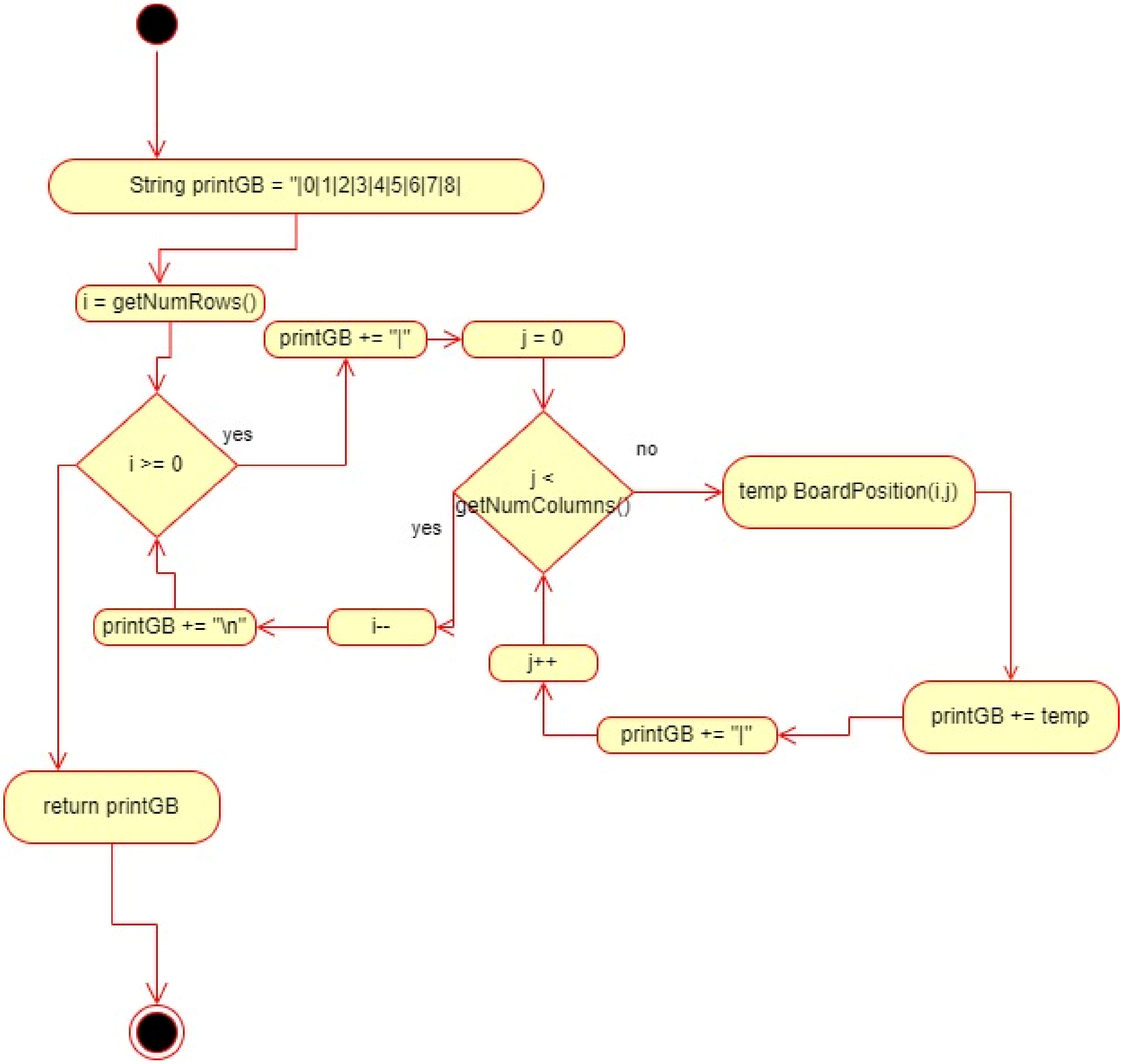
AbsGameBoard.java

Class Diagram



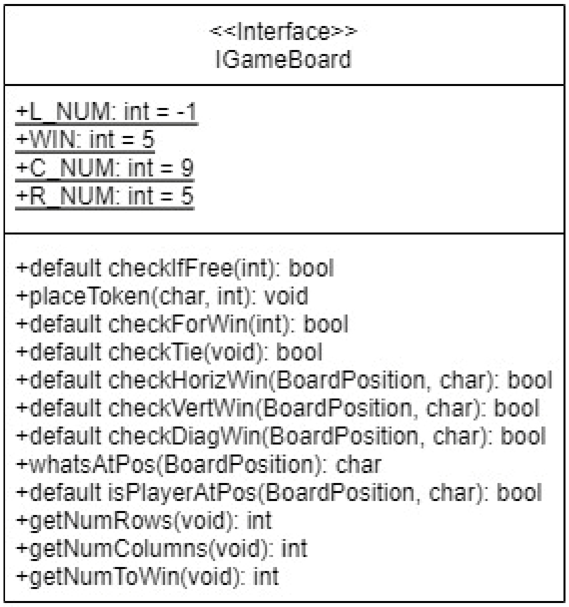
Activity Diagram

public string toString(GameBoard board)



IGameBoard

Class Diagram

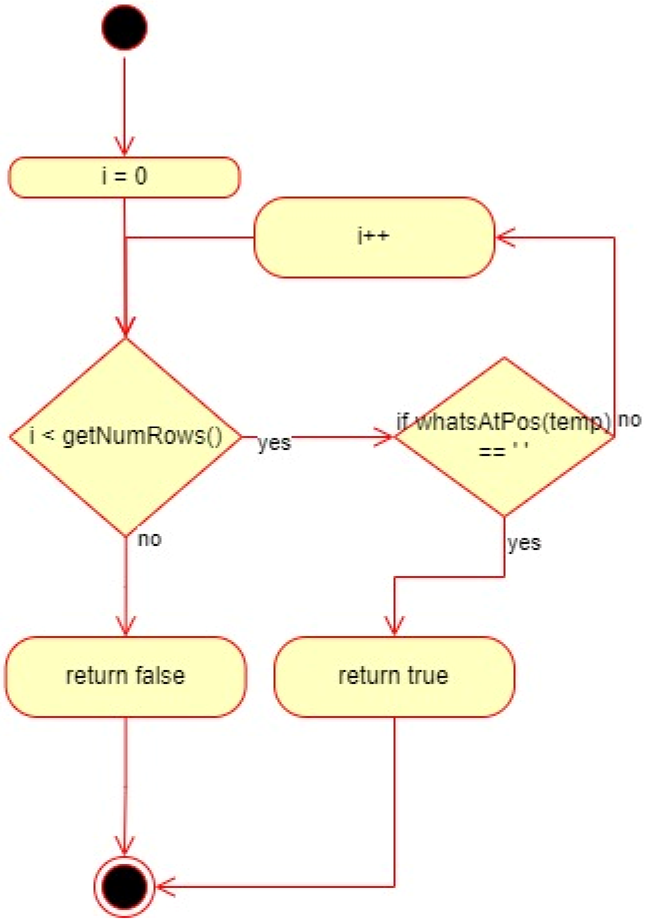


<<Interface>>
IGameBoard

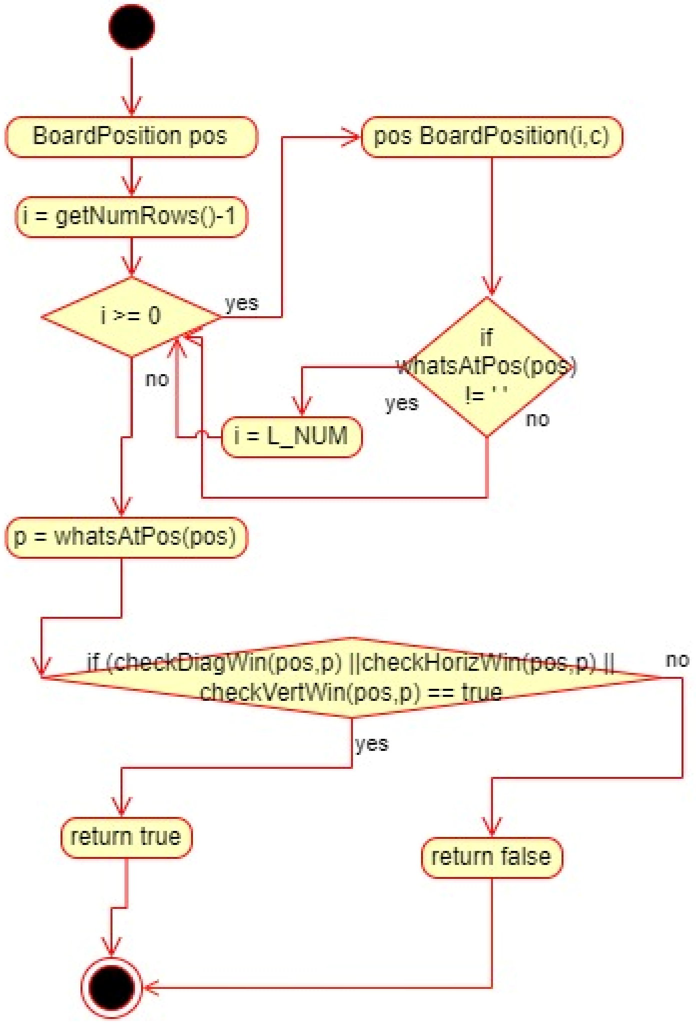
+L_NUM: int = -1
+WIN: int = 5
+C_NUM: int = 9
+R_NUM: int = 5

+default checkIfFree(int): bool
+placeToken(char, int): void
+default checkForWin(int): bool
+default checkTie(void): bool
+default checkHorizWin(BoardPosition, char): bool
+default checkVertWin(BoardPosition, char): bool
+default checkDiagWin(BoardPosition, char): bool
+whatsAtPos(BoardPosition): char
+default isPlayerAtPos(BoardPosition, char): bool
+getNumRows(void): int
+getNumColumns(void): int
+getNumToWin(void): int

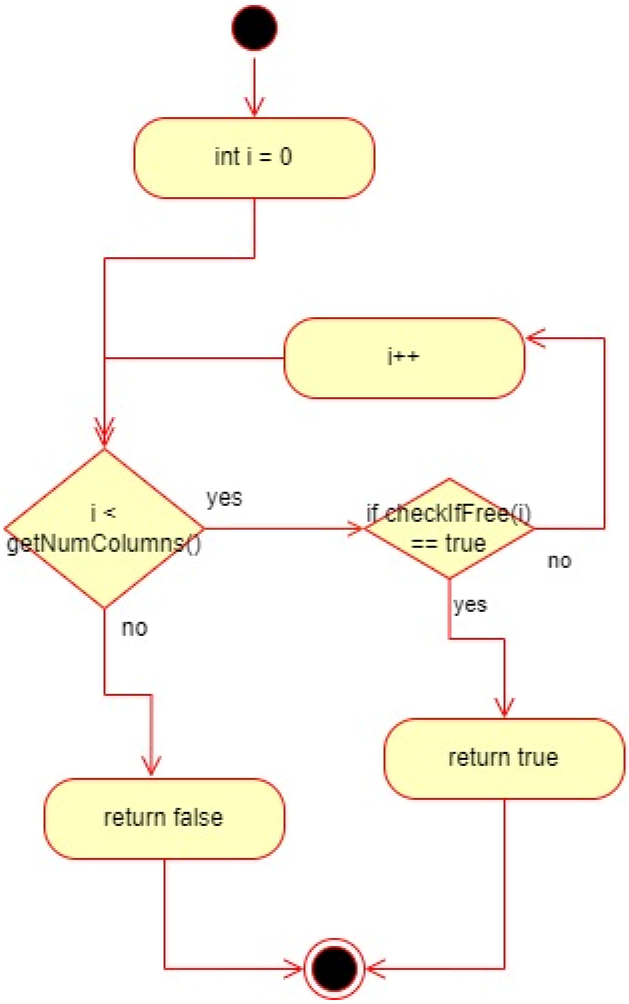
default public boolean checkIfFree(int c)



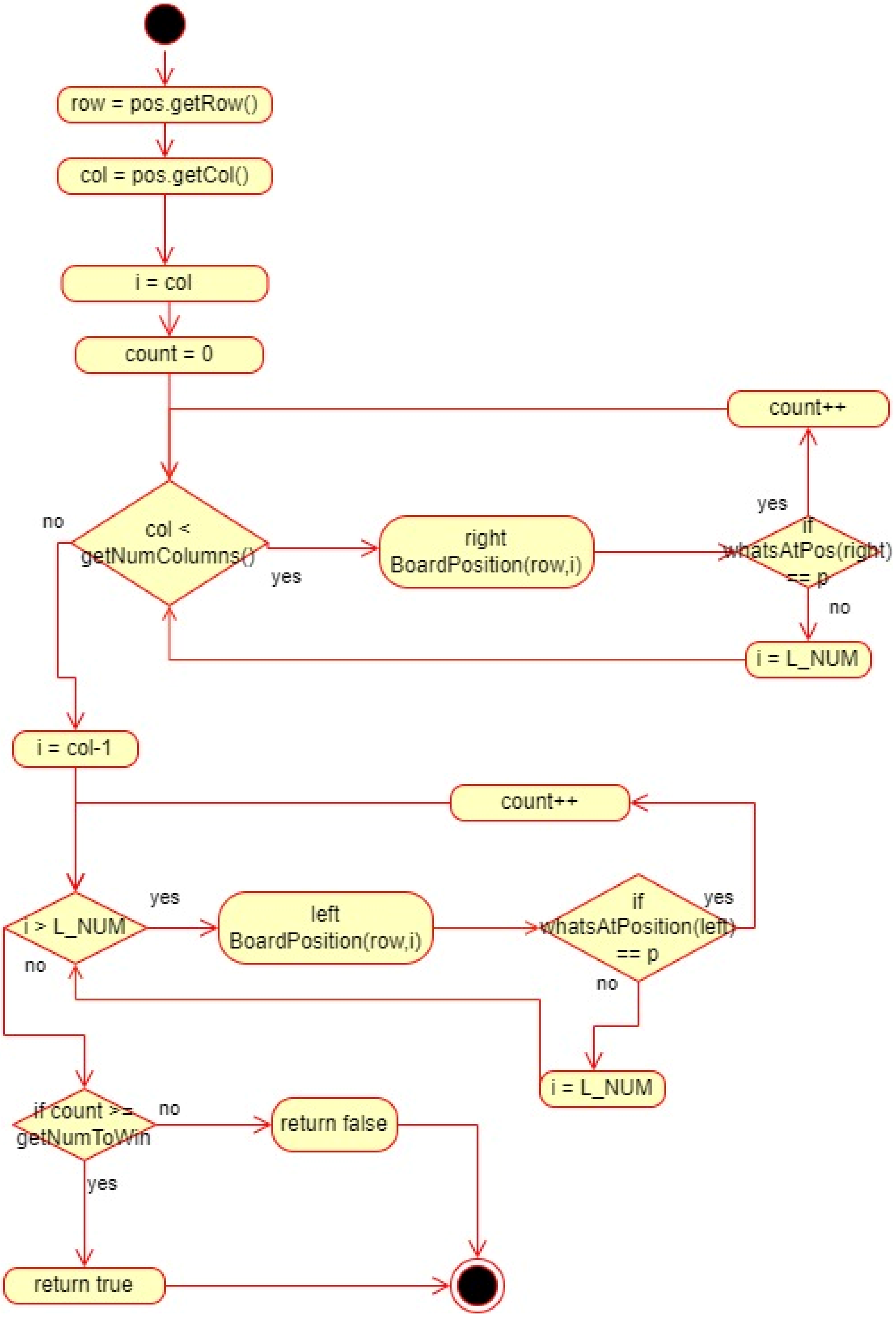
default public boolean checkForWin(int c)



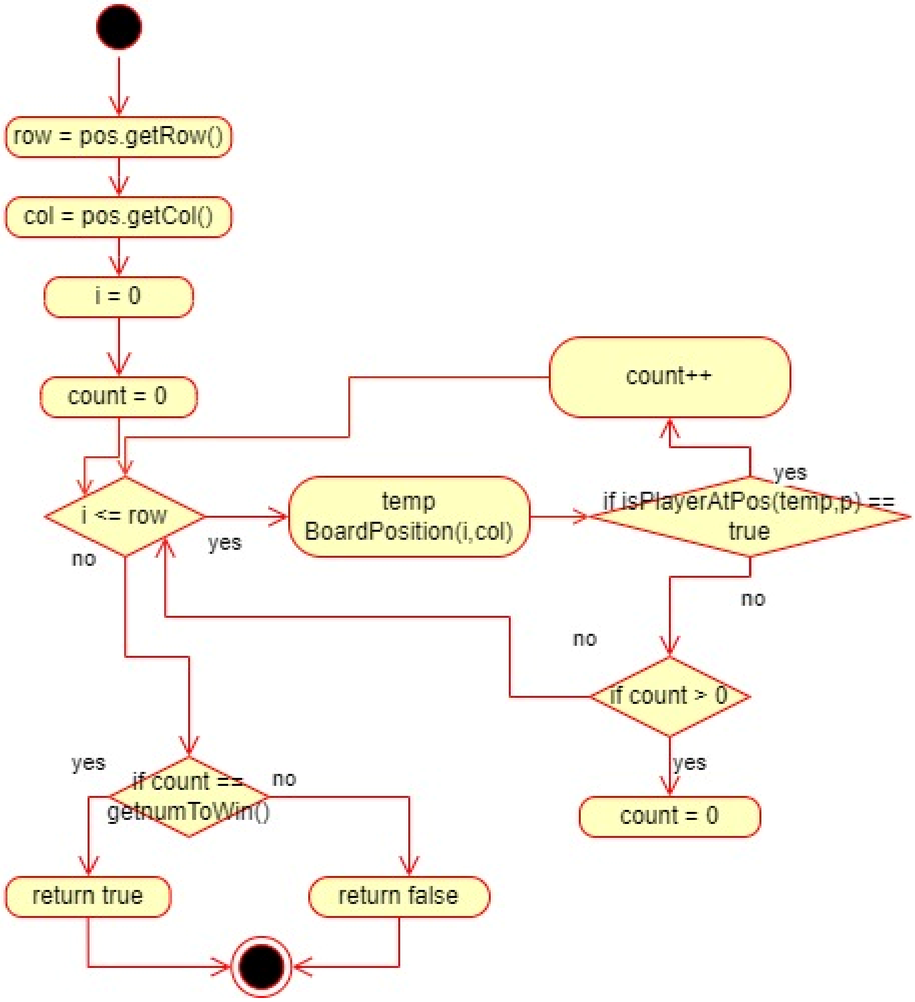
default public boolean checkTie()



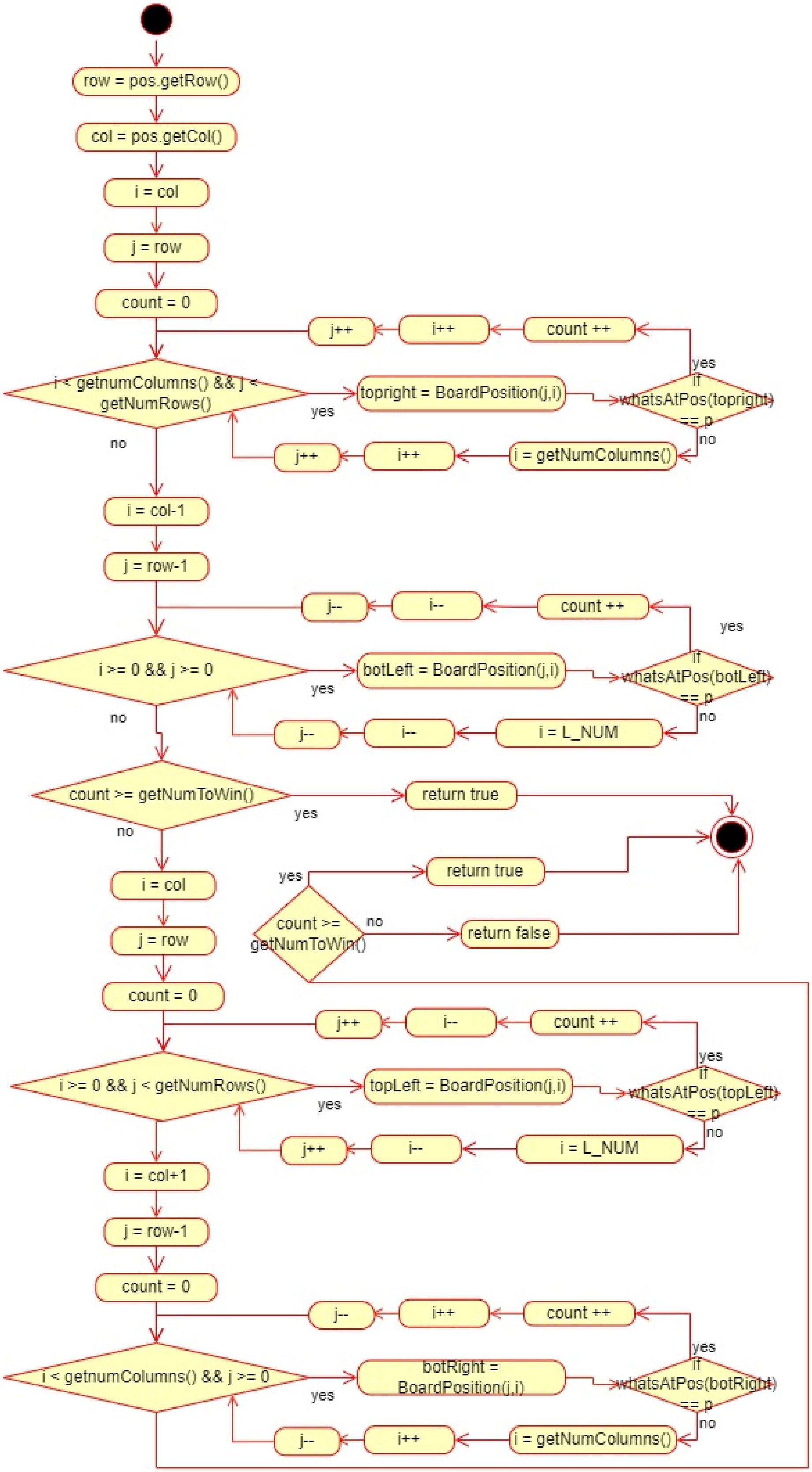
default public boolean checkHorizWin(BoardPosition pos, char p)

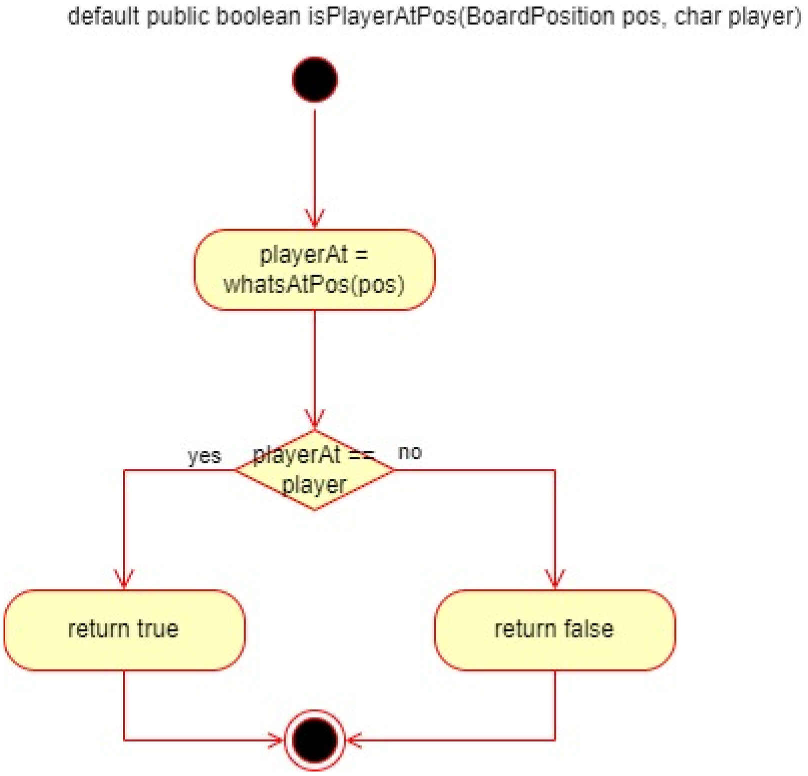


default public boolean checkVertWin(BoardPosition pos, char p)



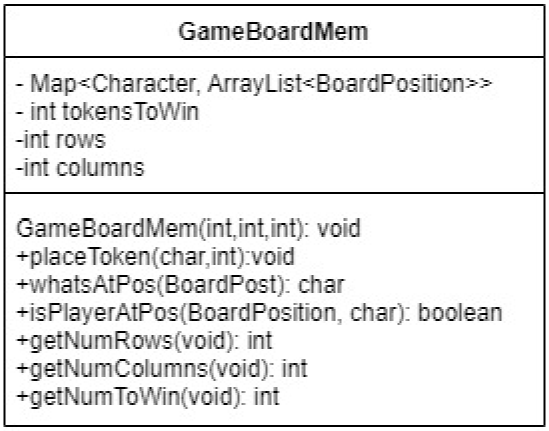
default public boolean checkDiagWin(BoardPosition pos, char p)





GameBoardMem.java

Class Diagram



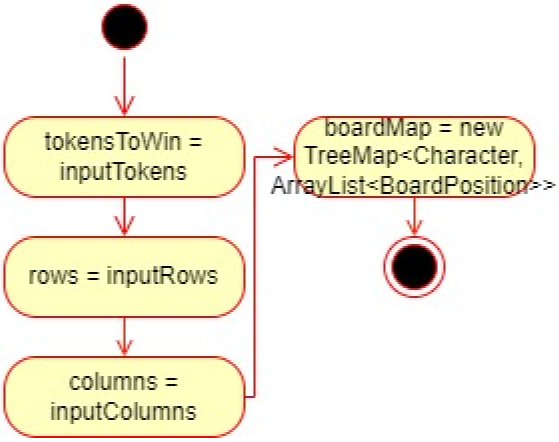
GameBoardMem

- Map<Character, ArrayList<BoardPosition>>
- int tokensToWin
-int rows
-int columns

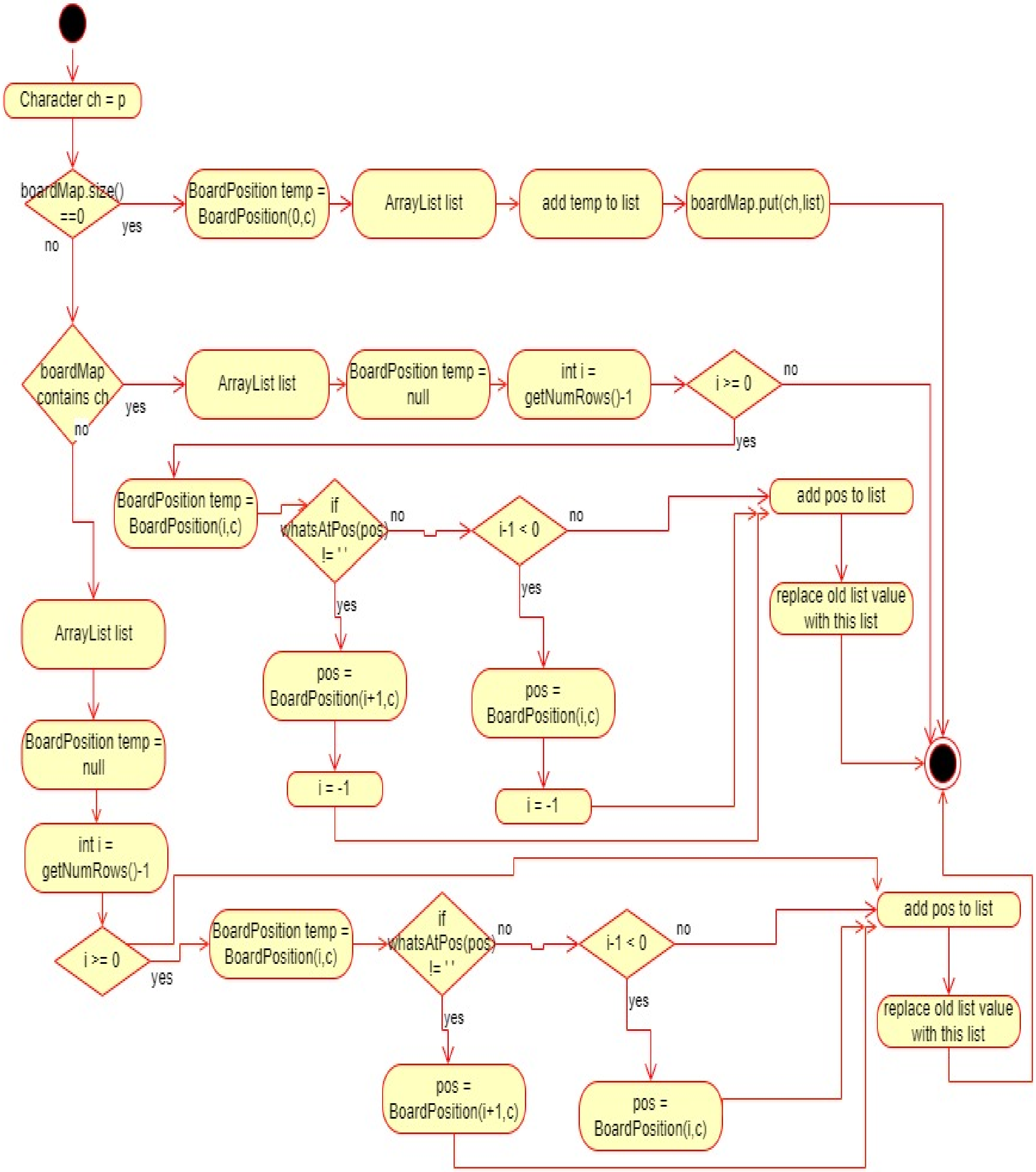
GameBoardMem(int,int,int): void
+placeToken(char,int):void
+whatsAtPos(BoardPost): char
+isPlayerAtPos(BoardPosition, char): boolean
+getNumRows(void): int
+getNumColumns(void): int
+getNumToWin(void): int

Activity Diagram

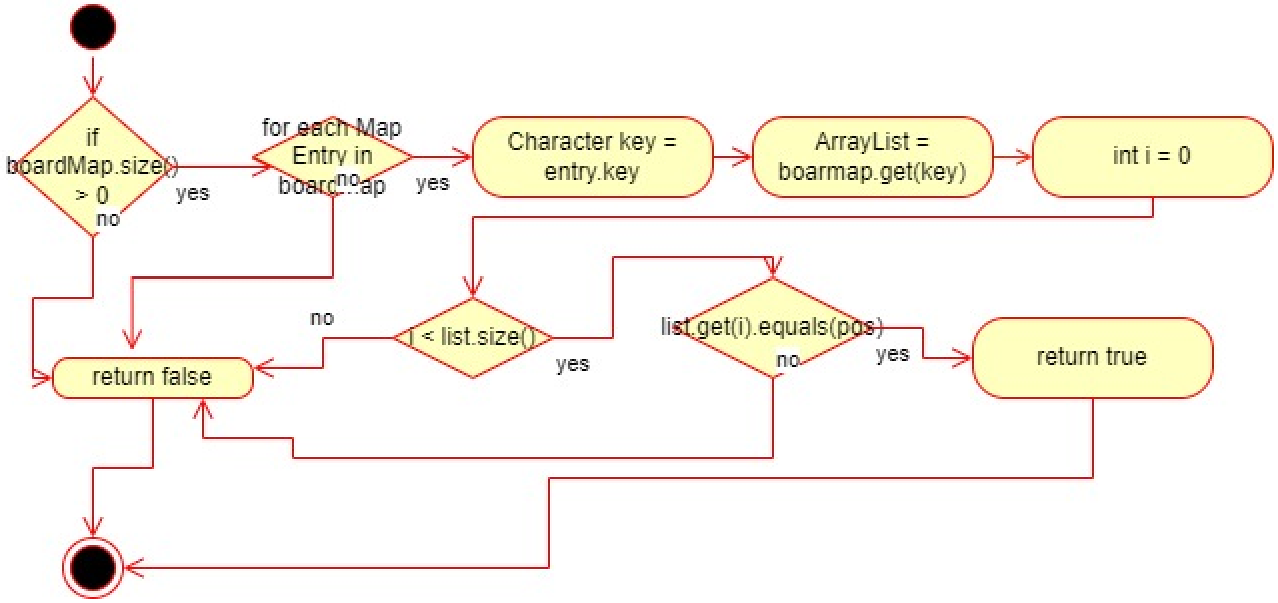
GameBoardMem(int inputTokens, int inputRows, int inputColumns)


tokensToWin = inputTokens
rows = inputRows
columns = inputColumns
boardMap = new TreeMap<Character, ArrayList<BoardPosition>>

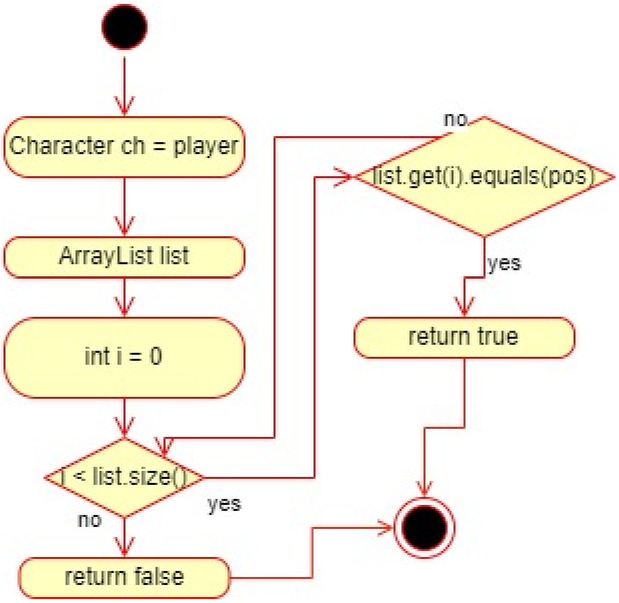
placeToken(char p, int c)

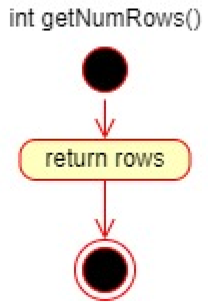

Character ch = p
boardMap.size() ==0
yes
no
BoardPosition temp = BoardPosition(0,c)
ArrayList list
add temp to list
boardMap.put(ch,list)
boardMap contains ch
yes
no
ArrayList list
BoardPosition temp = null
int i = getNumRows()-1
i >= 0
no
yes
BoardPosition temp = BoardPosition(i,c)
if whatsAtPos(pos) != ' '
no
i-1 < 0
no
add pos to list
yes
yes
replace old list value with this list
ArrayList list
pos = BoardPosition(i+1,c)
pos = BoardPosition(i,c)
i = -1
i = -1
BoardPosition temp = null
int i = getNumRows()-1
add pos to list
i >= 0
yes
BoardPosition temp = BoardPosition(i,c)
if whatsAtPos(pos) != ' '
no
i-1 < 0
no
yes
yes
replace old list value with this list
pos = BoardPosition(i+1,c)
pos = BoardPosition(i,c)

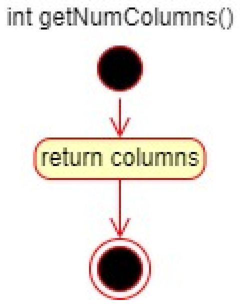
char whatsAtPos(BoardPosition pos)

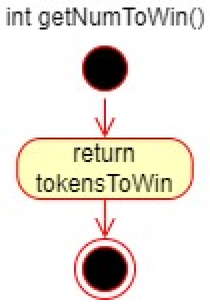


boolean isPlayerAtPos(BoardPosition pos, char player)









Test Cases

Details in Project 4.