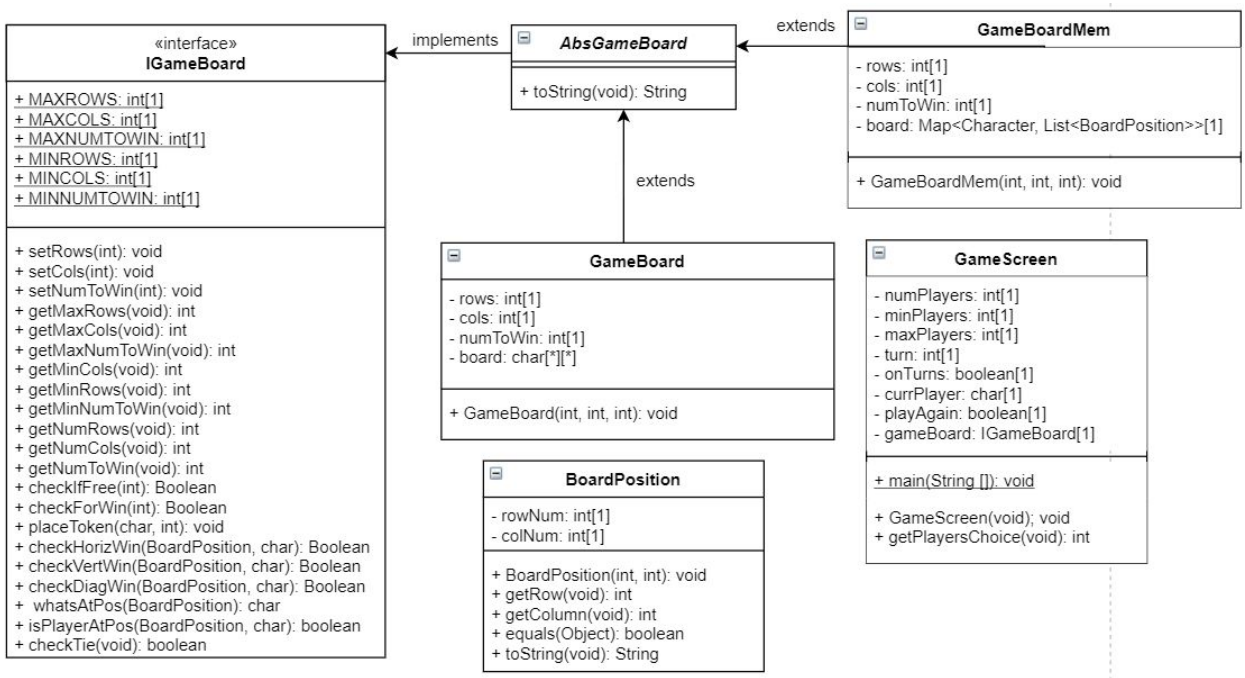
Requirements Analysis:

- Functional Requirements:
  - As a player, I must be able to see the board so that I know the current board state.
  - As a player, I must be able to choose the total number of players so that I can play with other people.
  - As a player, I must be able to choose my token so I know which tokens on the board are mine.
  - As a player, I must be able to choose an invalid token so I know which tokens are invalid.
  - As a player, I must be able to choose the number of rows in the board so I can make the game board the size of my choosing.
  - As a player, I must be able to choose an invalid number of rows so I know when the game board is getting too big or too small.
  - As a player, I must be able to choose the number of columns in the board so I can make the game board the size of my choosing.
  - As a player, I must be able to choose an invalid number of columns so I know when the game board is getting too big or too small.
  - As a player, I must be able to choose how many consecutive tokens are required to win so I can set how easily I can win the game.
  - As a player, I must be able to choose an invalid number of consecutive tokens required to win so I know when the game is too difficult or too easy to win.
  - As a player, I must be able to choose a speed-efficient gameboard so that I can play a fast game.
  - As a player, I must be able to choose a memory-efficient gameboard so that I can play a memory-efficient game.
  - As a player, I must be able to choose a column number to place my token.
  - As a player, I must be able to choose an invalid column so I know that I have made an illegal play.
  - As a player, I must be able to make my column choice after my opponents if they did not win so that I can continue playing the game.
  - As a player, I must be able to win horizontally in order to win.
  - As a player, I must be able to win vertically in order to win.
  - As a player, I must be able to win diagonally in order to win.
  - As a player, I must be able to see who won the game so that I know if I won or lost.
  - As a player, I must be able to see if the game is a tie, so that I know the game ended in a tie.
  - As a player, I must be able to choose whether or not I want to play again in case I want to play another game.
  - As a player, I must be able to choose new game settings whenever I start a new game in case I want to change any settings from the previous game.

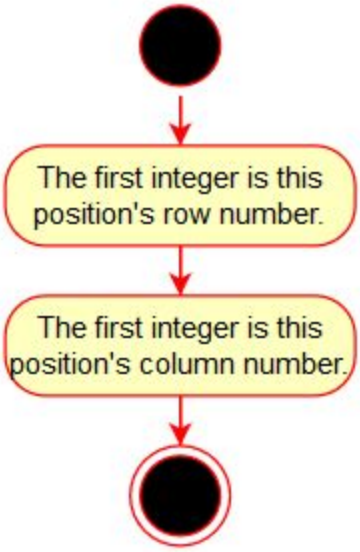
- ➢ Nonfunctional Requirements:
  - The game must be programmed using Java.
  - The game must run on a Unix system.
  - The bottom left corner of the board is (0, 0).
  - The maximum size of the board is 100 x 100.
  - The minimum size of the board is 3 x 3.
  - The size of the game board is determined by the player.
  - The maximum tokens in a row to win is 25.
  - The minimum tokens in a row to win is 3.
  - The number of tokens in a row to win is determined by the player.
  - The maximum number of players is 10.
  - The minimum number of players is 2.
  - No two players may choose the same token.
  - Player 1 always goes first.

Class Diagrams:

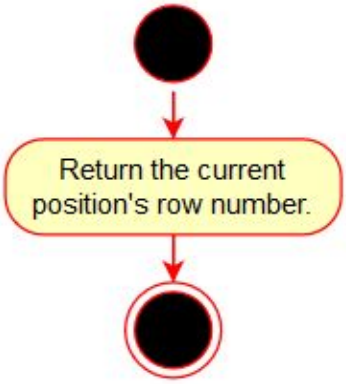


BoardPosition Class:

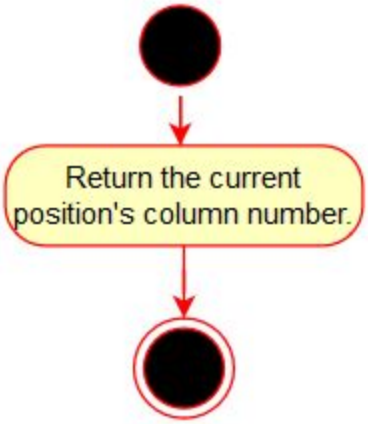
public BoardPosition(int row, int column):



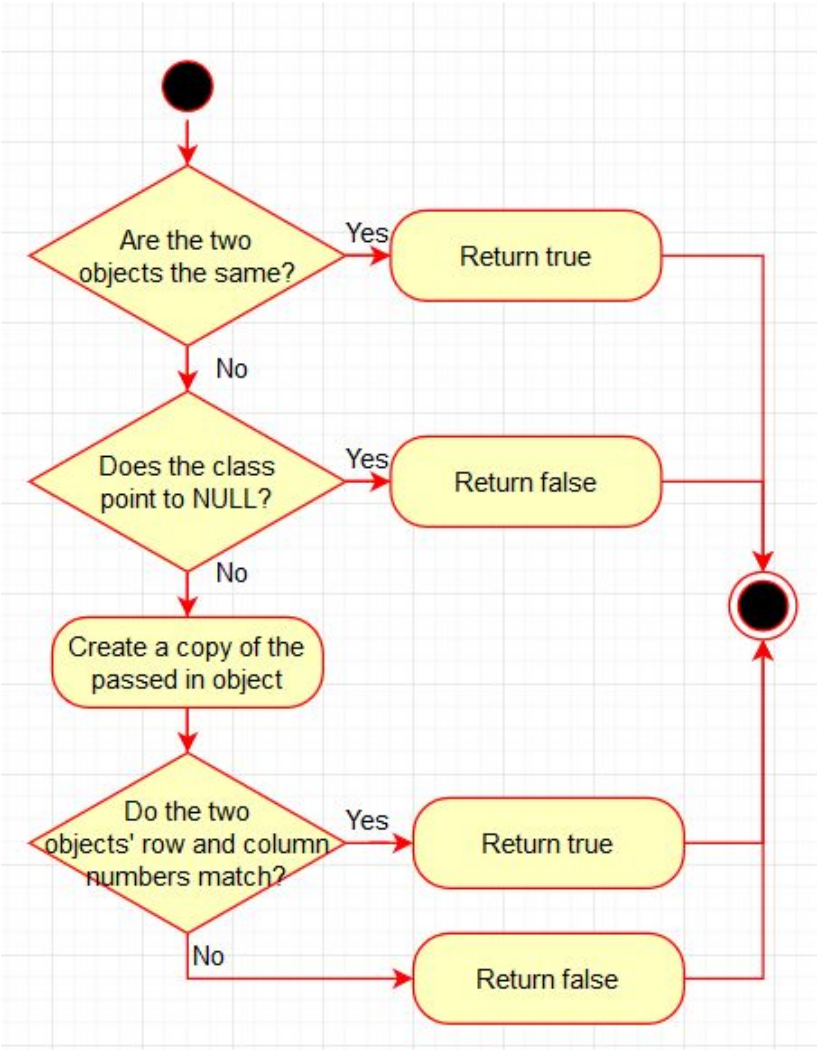
public int getRow():



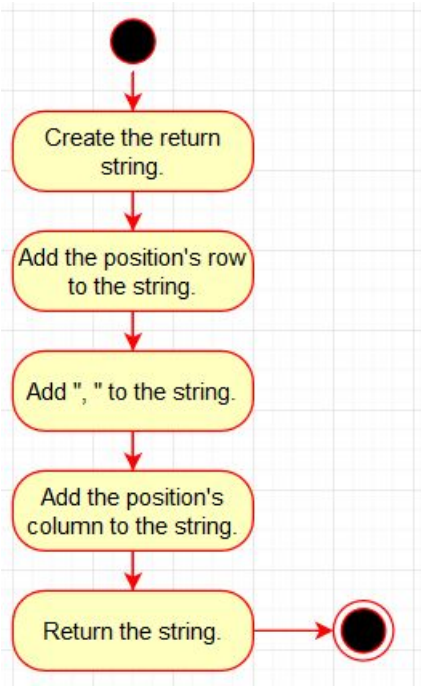
public int getColumn():



public bool equals():

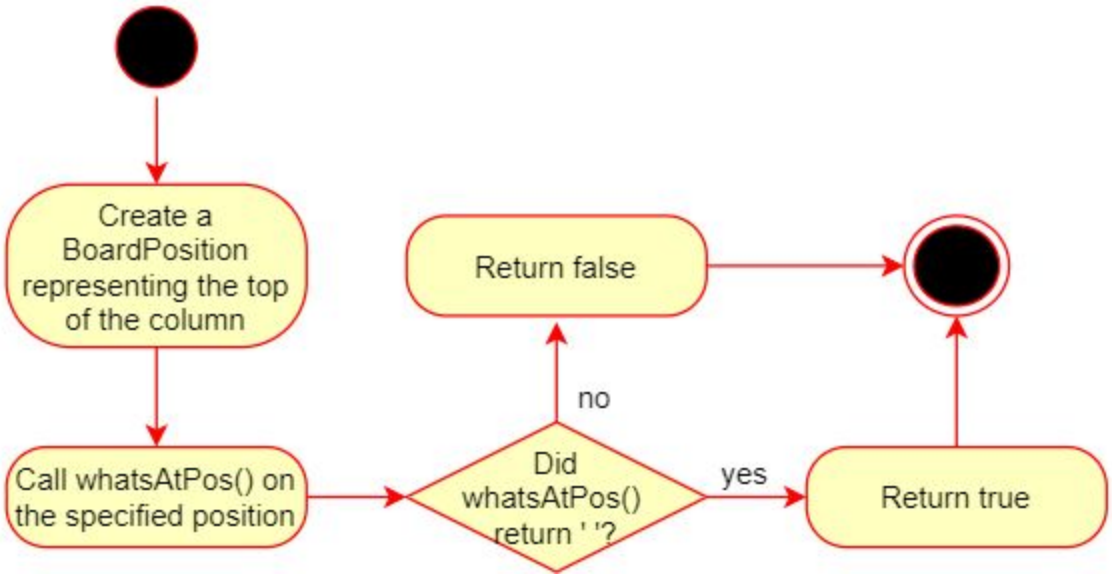


public String toString():

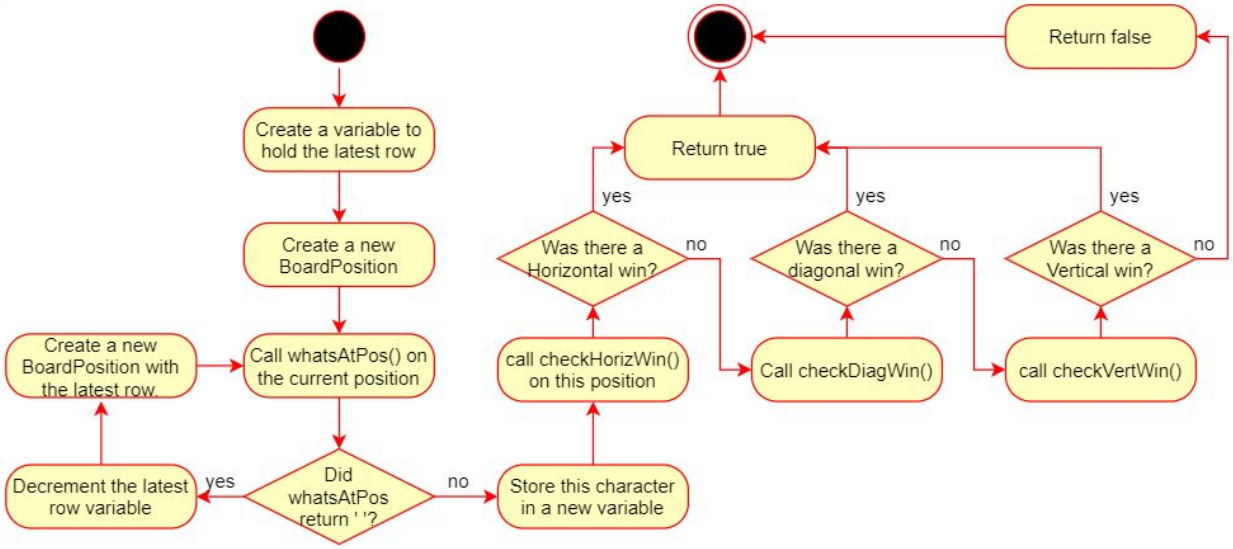


IGameBoard Interface:

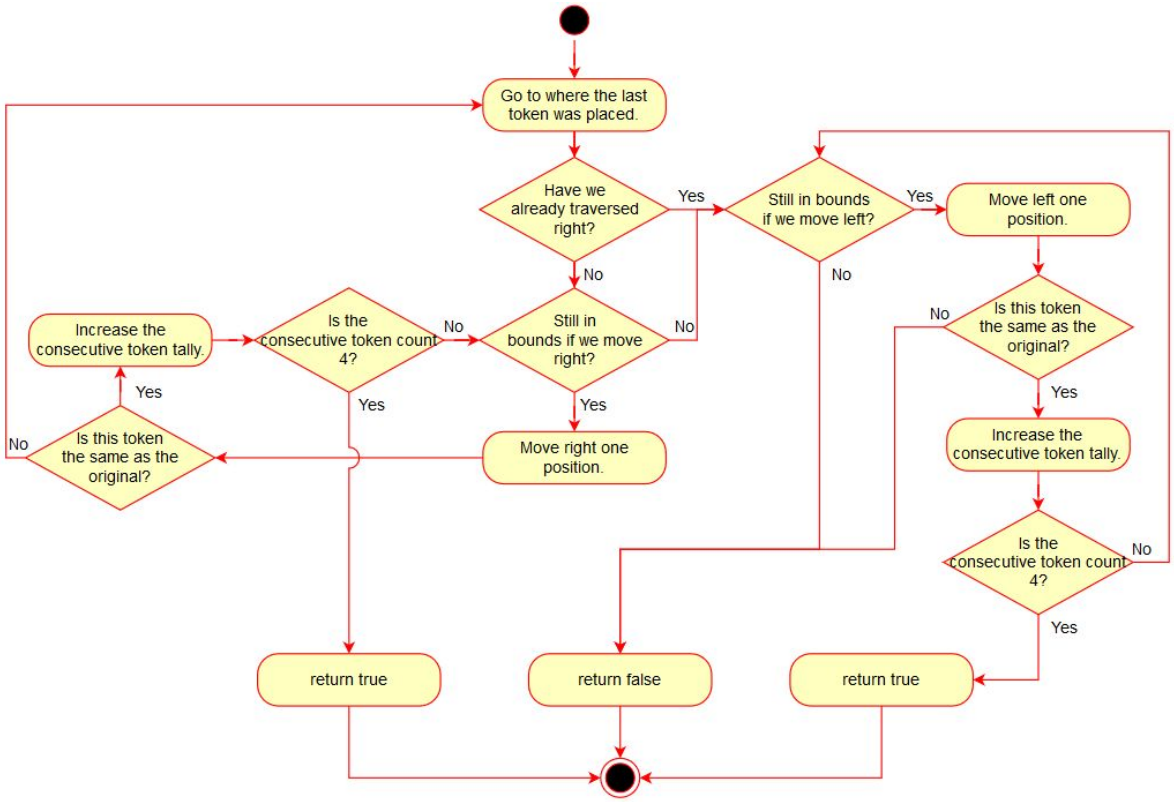
public boolean checkIfFree(int c):



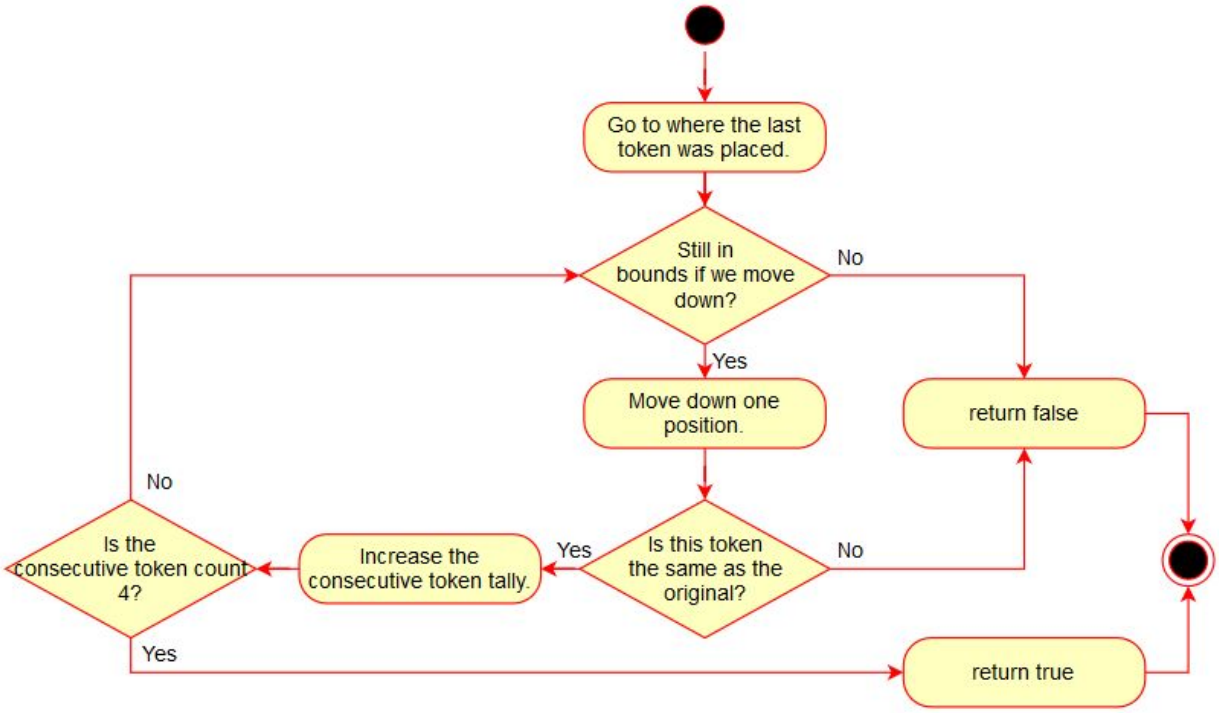
public boolean checkForWin(int c):



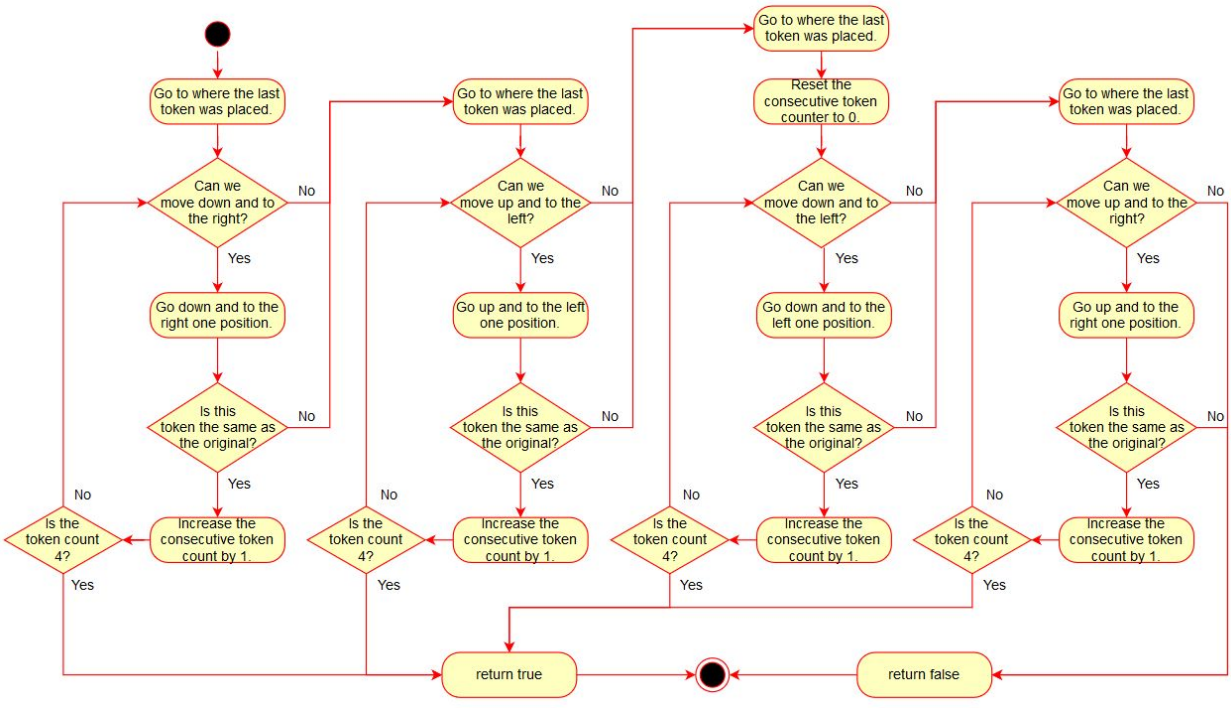
public boolean checkHorizWin(BoardPosition pos, char p):



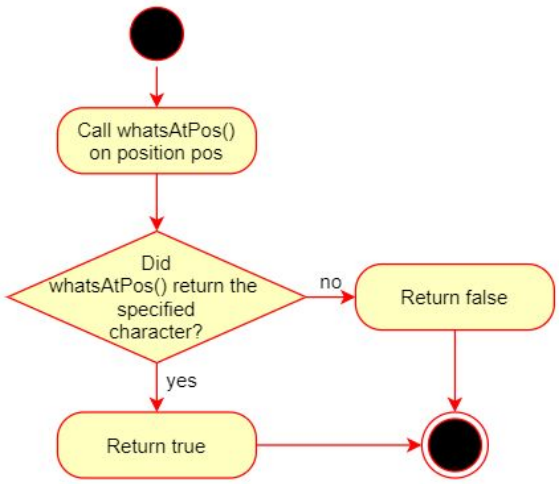
public boolean checkVertWin(BoardPosition pos, char p):



public boolean checkDiagWin(BoardPosition pos, char p):

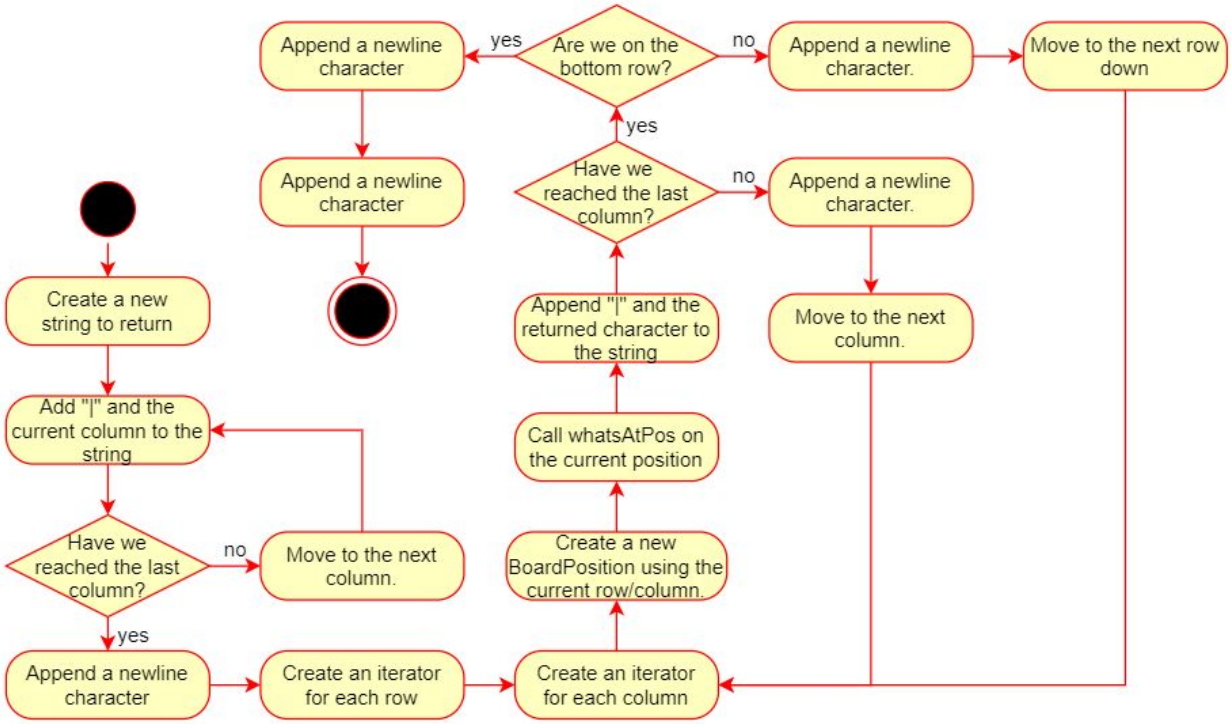


public boolean isPlayerAtPos(BoardPosition pos):



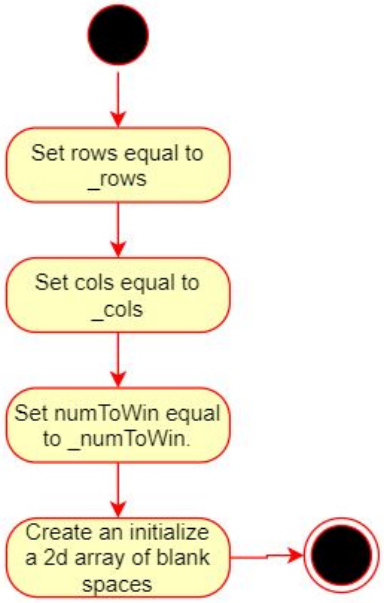
AbsGameBoard:

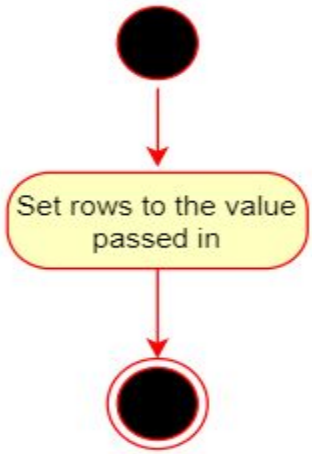
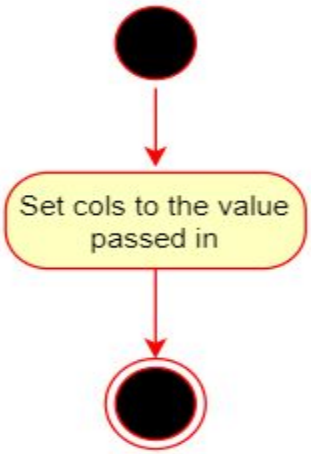
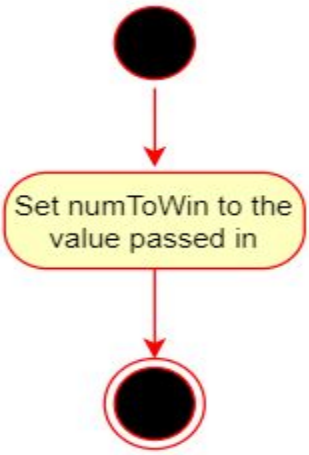
String toString():

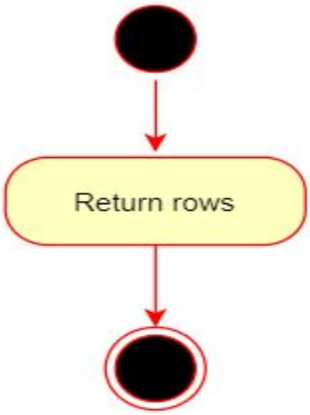
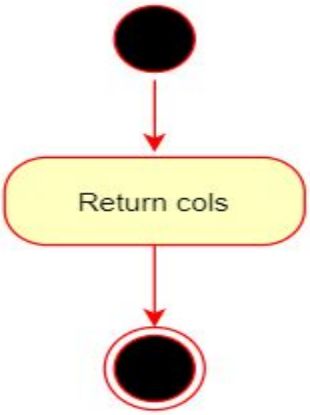
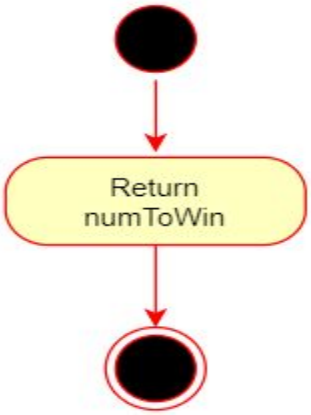


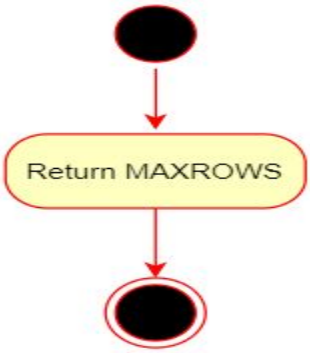
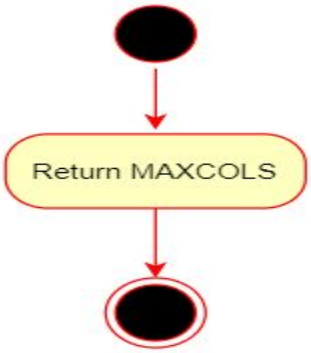
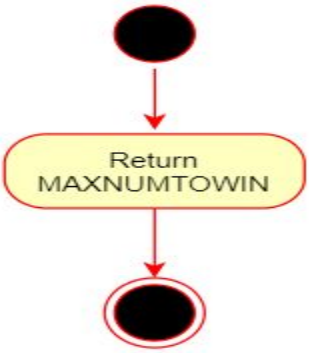
GameBoard:

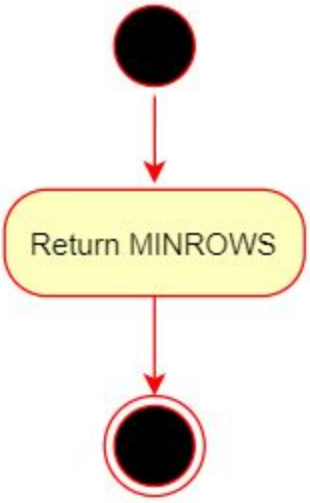
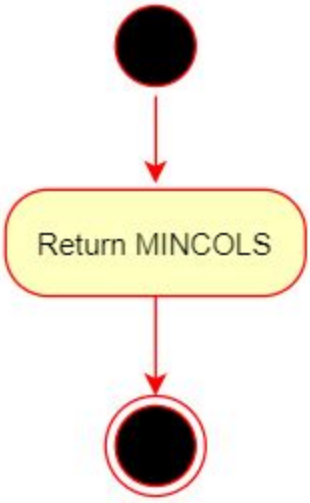
public GameBoard():



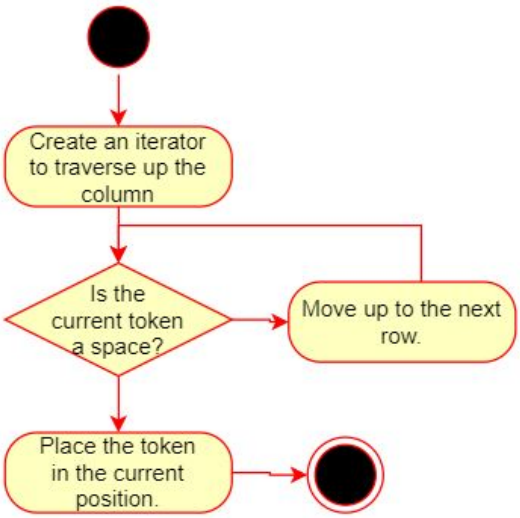
Void setRows()	Void setCols()	Void setNumToWin()
		

Int getNumRows()	Int getNumCols()	Int getNumToWin()
		

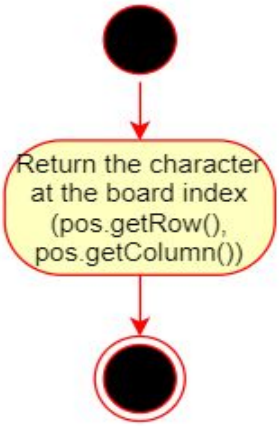
Int getMaxRows()	Int getMaxCols()	Int getMaxNumToWin()
		

Int getMinRows()	Int getMinCols()	Int getMinNumToWin()
		

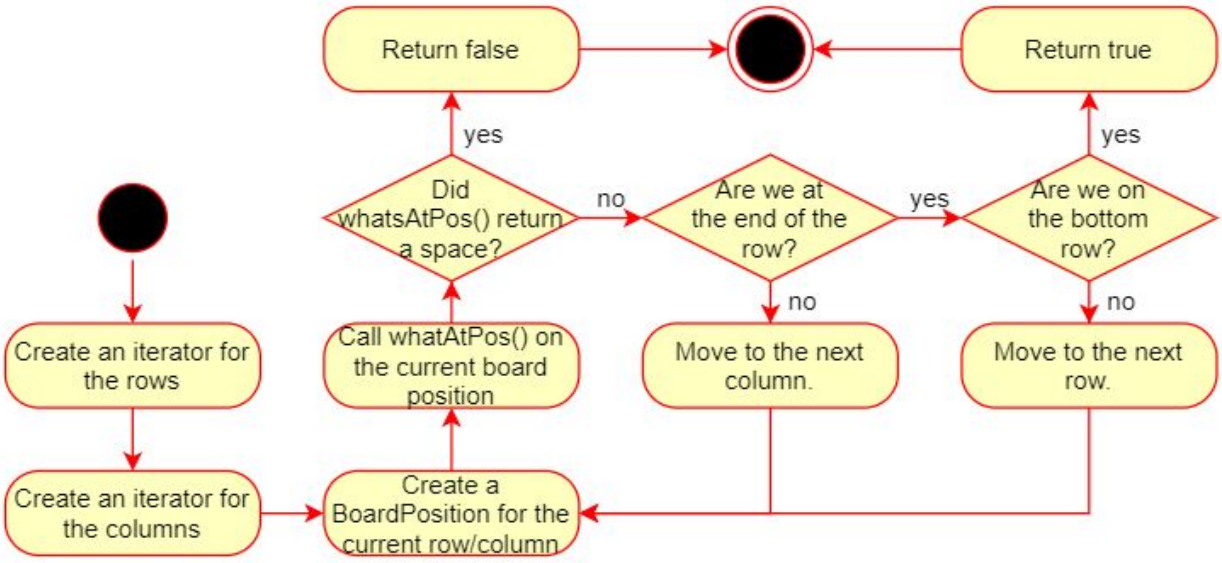
Void placeToken():



Char whatsAtPos():

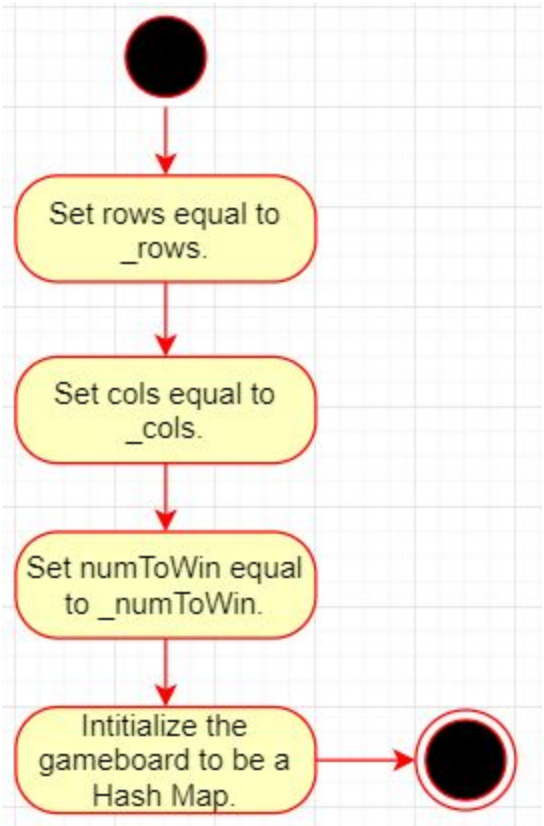


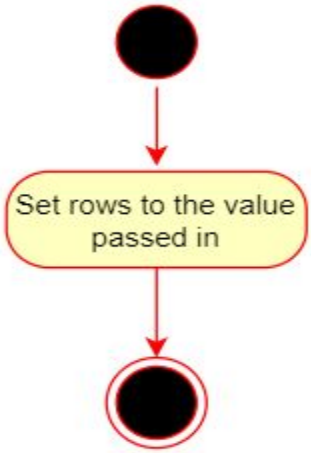
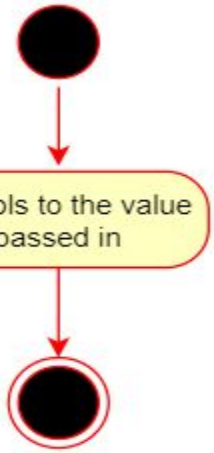
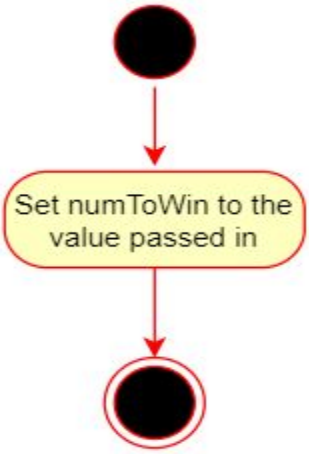
boolean checkTie():

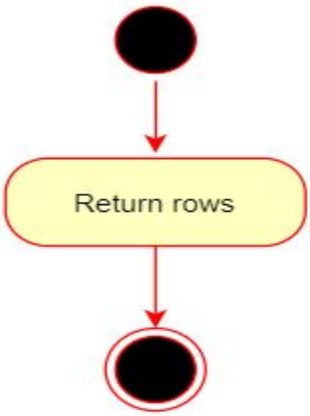
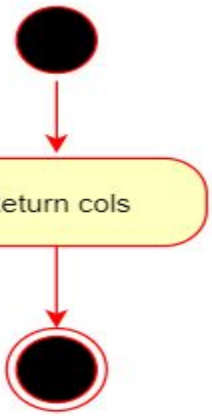
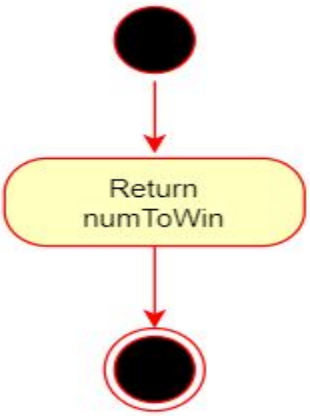


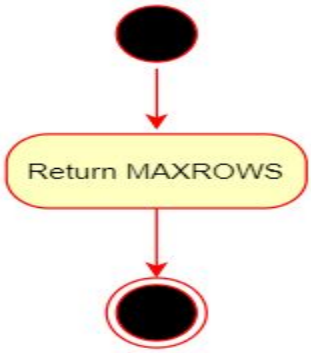
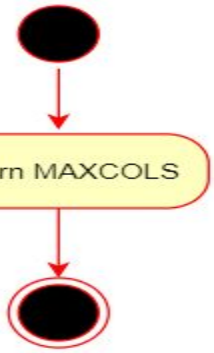
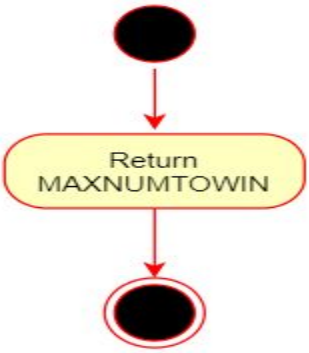
GameBoardMem:

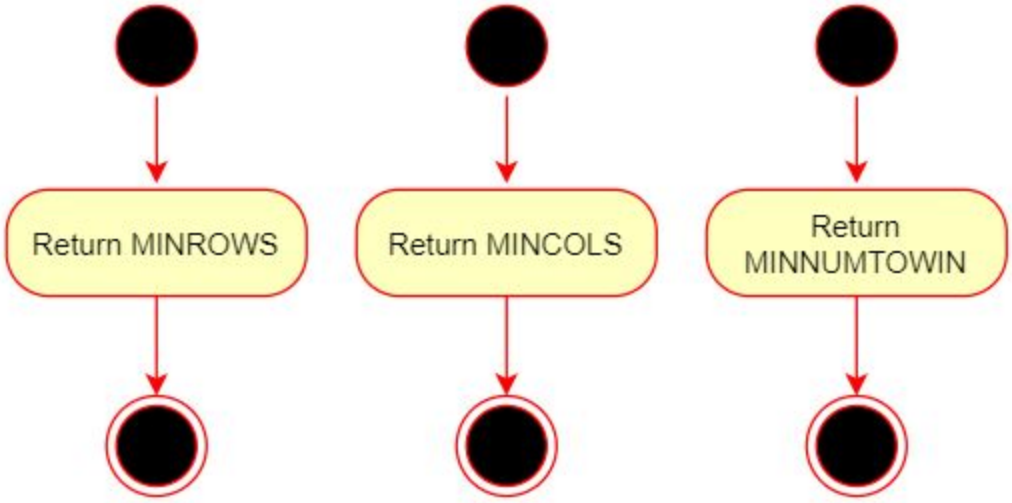
Void GameBoardMem:



Void setRows()	Void setCols()	Void setNumToWin()
		

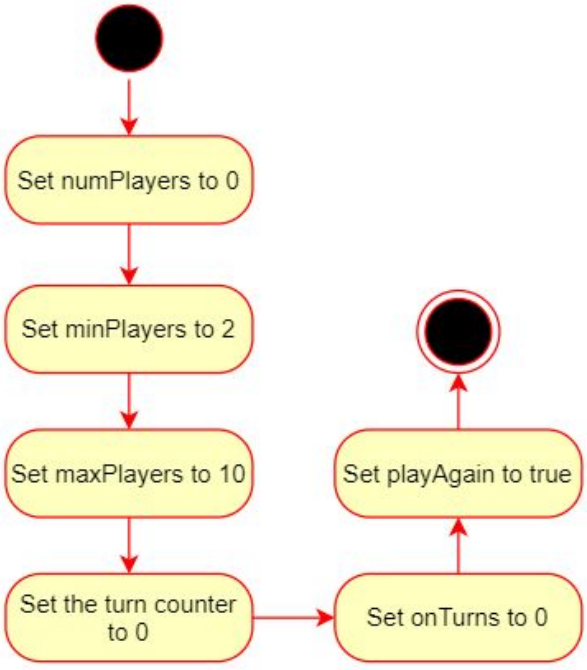
Int getNumRows()	Int getNumCols()	Int getNumToWin()
		

Int getMaxRows()	Int getMaxCols()	Int getMaxNumToWin()
		

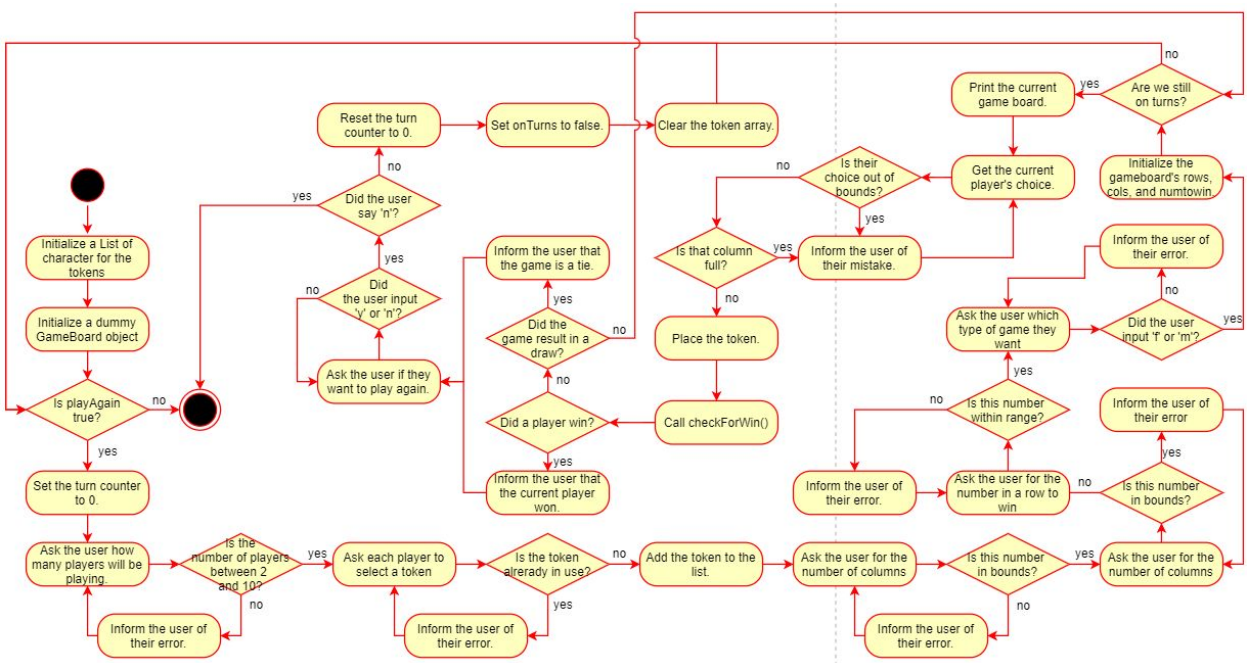
Int getMinRows()	Int getMinCols()	Int getMinNumToWin()
		

GameScreen:

public GameScreen():



public static void main(String [] args):



public int getPlayersChoice():