CPSC 2150 Project 1

Connect 4

George Jubenvill, Jake Barz, Haagen Williams

Requirements Analysis

Functional Requirements:

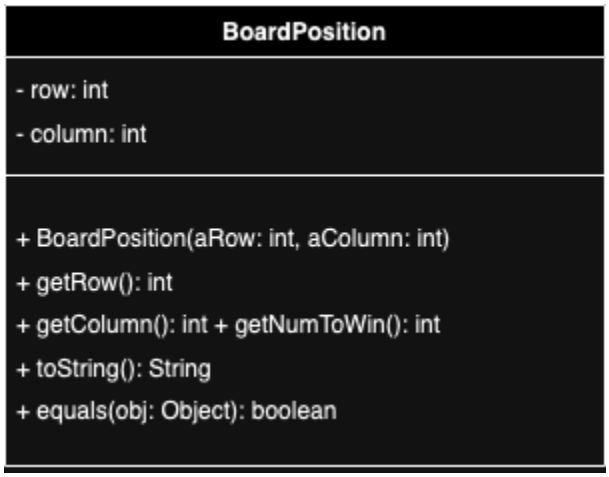
1. As a player, I can put pieces into the board so I can play the game.
2. As a player, I can input a column I wish to put my pieces in so that I can try to win.
3. As a player, I can choose to play again when the game is over so that I can keep playing the game.
4. As a player, I can place a piece after each of my opponent's placements so that the game is fair.
5. As a player, I can input a new column if the first choice was invalid so that I can choose a valid column without losing my turn.
6. As a player, I can make another choice if I initially choose a column that is full so that my turn does not get wasted.
7. As a player, I can visually see where all the pieces are on the board so that I can choose a good column for my pieces.
8. As a player, I can win by connecting pieces vertically, horizontally, or diagonally so that I can win the game.
9. As a player, I want the game to detect a tie if all spaces are filled and no player has won so that the game can end fairly.
10. As a player, I can see whose turn it is so that I know when it's my turn to play.
11. As a player, I can customize the size of the board and what the win condition in order to give the player personal preferences
12. As a player, I can pick between if I want to use a memory-efficient or fast implementation upon starting the game
13. As a player, I can have a minimum of 2 players and a maximum of 10 players in the game

Non-Functional Requirements

1. There should be a visual representation of the board.
2. Game should end when someone wins or there is a tie.
3. Players should be told who won or if it was a tie.
4. The game should still be playable if the board size changes.
5. Players should be notified if they pick a column of the board that is full.
6. The game board should be empty at the start of each game.
7. The game should keep track of where players have put their pieces
8. The game must support two players.
9. Each players' pieces should be distinct to be able to tell them apart.
10. The pieces I place need to drop to the bottom of the board, on top of the last piece placed in that column, so the game follows real-world physics.
11. The game board must support to size up to 100x100 for players
12. The game must run on Linux and Windows.
13. The game board must be at least 3x3 with 3 tokens to win
14. The game board should have a fast implementation using a 2D char array
15. The game board should have a memory efficient implementation using a map
16. The game should be written in Java
17. The game should be a command line program

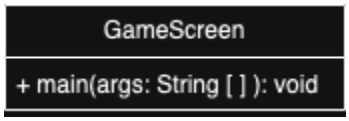
System Design – (UML diagrams)

BOARDPOSITION



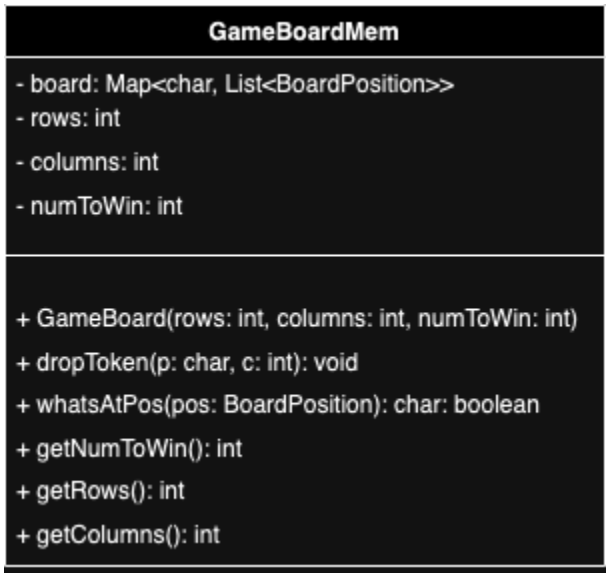
GAMEBOARD

GAMESCREEN

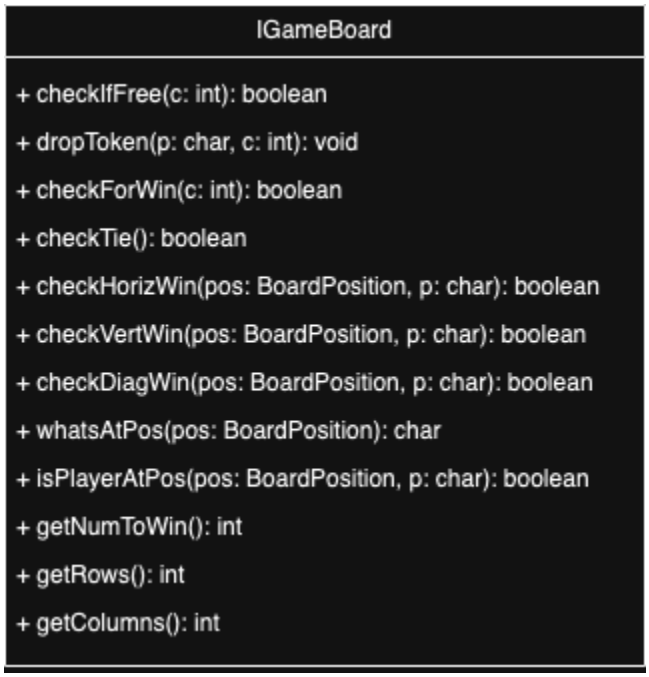


ABSGAMEBOARD

GAMEBOARDMEM



IGAMEBOARD


IGameBoard
+ checkIfFree(c: int): boolean
+ dropToken(p: char, c: int): void
+ checkForWin(c: int): boolean
+ checkTie(): boolean
+ checkHorizWin(pos: BoardPosition, p: char): boolean
+ checkVertWin(pos: BoardPosition, p: char): boolean
+ checkDiagWin(pos: BoardPosition, p: char): boolean
+ whatsAtPos(pos: BoardPosition): char
+ isPlayerAtPos(pos: BoardPosition, p: char): boolean
+ getNumToWin(): int
+ getRows(): int
+ getColumns(): int