CPSC 2150 Project 4 Report

Noah Fultz, James Moore, Reese Myers, and Bryan Pisciotti

Requirements Analysis

Functional Requirements:

1. As a player, I can select the number of players so that I can choose how many other players I will be playing with.
2. As a player, I can select a character to represent my token so that I can keep track of where I've placed my token
3. As a player, I am re-prompted if I select a token to represent me that has already been chosen by another player so that I can ensure that a unique token represents me.
4. As a player, I can choose to play a fast game so that the game application executes more quickly.
5. As a player, I can select the number of rows in the board so that I can determine the height of the board.
6. As a player, I am re-prompted if I select a number of rows that is too low so that I can select a number of rows that is large enough for the game to work as intended.
7. As a player, I am re-prompted if I select a number of columns that is too low so that I can select a number of columns that is large enough for the game to work as intended.
8. As a player, I am re-prompted if I select a number of tokens to win that is too low so that I can select a number of tokens to win that is large enough for the game to work as intended.
9. As a player, I am re-prompted if I select a number of players that is too low so that I ensure that I am playing the game with at least one other person.
10. As a player, I am re-prompted if I select a number of players that is too high so that I can select a number of players small enough for the game to work as intended.
11. As a player, I am re-prompted if I select a number of rows that is too high so that I can select a number of rows that is small enough for the game to work as intended.
12. As a player, I am re-prompted if I select a number of columns that is too high so that I can select a number of columns that is small enough for the game to work as intended.
13. As a player, I am re-prompted if I select a number of tokens to win that is too high so that I can select a number of tokens to win that is small enough for the game to work as intended.
14. As a player, I can select the number of columns in the board so that I can determine the width of the board.
15. As a player, I can select the number of tokens in a row required to win so that I can determine the requirement for winning.

16. As a player, I can choose to play a memory efficient game by typing ‘m’ or ‘M’ so that the game application uses less memory resources.
17. As a player, I am re-prompted to choose a fast or memory efficient game if I do not type ‘m’,’M’, ‘f’, or ‘F’, so that I ensure that I choose one of the two.
18. As a player, I can view whose turn it is prior to a move so that I know if it is my turn to move.
19. As a player, I am prevented from selecting a full column so that I cannot make an illegal move.
20. As a player, I am re-prompted if I select a non-existent column number so I am able to provide corrected input and make a legal move.
21. As a player, I am re-prompted if I drop my token in a full row so that I am able to provide corrected input and make a legal move.
22. As a player, I am able to win by placing a number of my tokens equal to the amount I selected to require to win on top of each other so that I can win vertically.
23. As a player, I am able to win by placing a number of my tokens equal to the amount I selected to require to win side by side so that I can win horizontally.
24. As a player, I am able to win by lining up a number of my tokens equal to the amount I selected to require to win diagonally so that I can win diagonally.
25. As a player, I am informed of a winner when appropriate so that I know who won and that the game is over.
26. As a player, I am informed of a tie when appropriate so that I know that there is no winner and the game is over.
27. As a player, I can choose which column to place my token in so that I can play the game.
28. As a player, I am shown the board after my own turn so that I can verify my move.
29. As a player, I am shown the board after my opponent’s turn so that I can make an informed decision regarding where to move.
30. As a player, I can choose to play another game after a game’s conclusion so that I can play connectx again without restarting the program.
31. As a player, I can view the board after winning or losing to see where the required number of tokens were placed in a row successfully.
32. As a player, I can view the board after a tie so that I know that the board is completely full and no one has won.
33. As a player, I can view column labels so that I can easily determine column numbers.
34. As a player, I can **c**hoose to not play another game after a game ends so that the program will end.
35. As a player, I can change the number of players if I choose to play another game so that I can play with a different number of players.
36. As a player, I can choose a different token to represent me if I choose to play another game so that I can use a different token in the board
37. As a player, I can change the number of rows if I choose to play another game so that I can play another game with a board of a different height.
38. As a player, I can change the number of columns if I choose to play another game so that I can play another game with a board of a different width.

39. As a player, I can change the number of tokens in a row required to win if I play another game so that I can alter the requirements to win.
40. As a player, I can select either the fast or memory-efficient option again if I play another game so that I can make the game be either faster or more memory efficient.

Non-Functional Requirements

1. The program must run on a command-line interface.
2. The program must be modular to enable future flexibility.
3. The program must make use of constants.
4. The program must be fast enough to be playable.
5. The program must be written in Java.
6. The program must be able to handle illegal moves without failing.
7. The program must be able to switch turns appropriately.
8. The program must ensure that dropped tokens fall to the lowest empty row.
9. The program must be able to actively keep track of the state of each board position.
10. The program must allow for a maximum of 100 rows in the board.
11. The program must allow for a maximum of 100 columns in the board.
12. The program must require at least 3 rows in the board.
13. The program must require at least 3 columns in the board.
14. The program must require at least 3 tokens in a row to win.
15. The program must require that the number of tokens to win is less than both the row and column values.
16. The program must require that the number of tokens to win is less than or equal to 25
17. The program must require that there are at least 2 players.
18. The program must require that there are no more than 10 players
19. The program must include an implementation that utilizes a 2D array to represent the board.
20. The program must include an implementation that utilizes a hash map to represent the board.
21. The program must convert letters to uppercase to use as tokens.
22. The program must be designed around the principle of coding to the interface.
23. The program must be designed such that turns occur in the order that players enter their desired tokens.
24. The board must be able to display both 1 and 2 digit column labels.
25. The program must be designed such that position [0][0] represents the bottom left of the board
26. The constructors for objects representing game boards should take parameters in the order of rows, columns, and number of tokens needed to win.

Deployment Instructions

To compile the program, issue the “make” command from the terminal within the root directory of the project.

To run the program, ensure that the “make” command has already been issued, then issue the “make run” command from within the same directory.

To compile the test cases, issue the “make test” command from the root directory of the project.

To test GameBoard, ensure that the “make test” command has already been issued, then issue the “testGB” command from within the same directory.

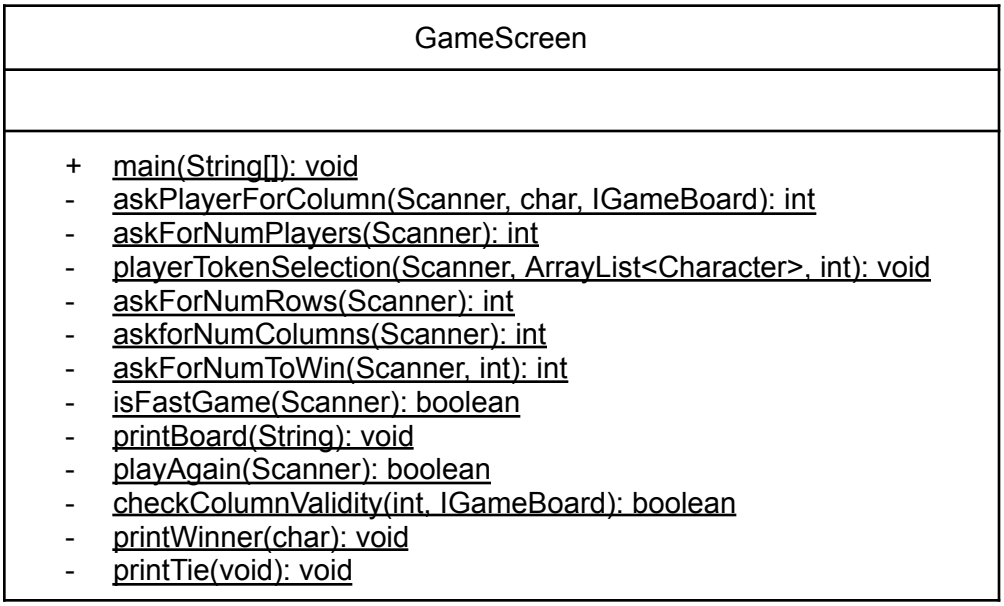
To test GameBoardMem, ensure that the “make test” command has already been issued, then issue the “testGBMem” command from within the same directory.

To delete all class files that have been generated during compilation, issue the “make clean” command from the root directory of the project

System Design

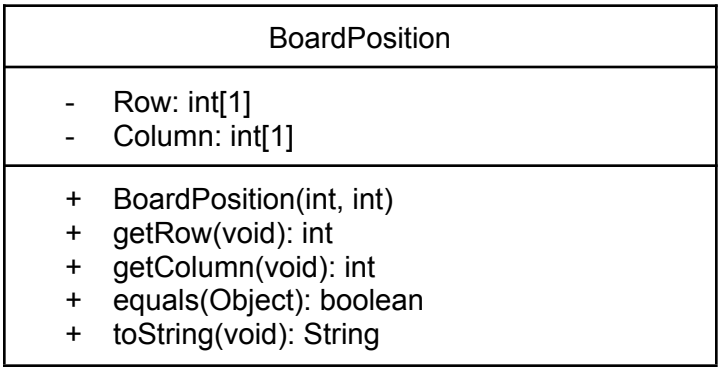
GameScreen:

Class diagram



BoardPosition:

Class diagram



Class Diagram: (IGameBoard ,AbsGameBoard, GameBoardMem)