CPSC 2150 Project 3 Report

Abby Clanton, Cassie Philips, Ricardo Varona, Kaki Newsome

Requirements Analysis

Functional Requirements:

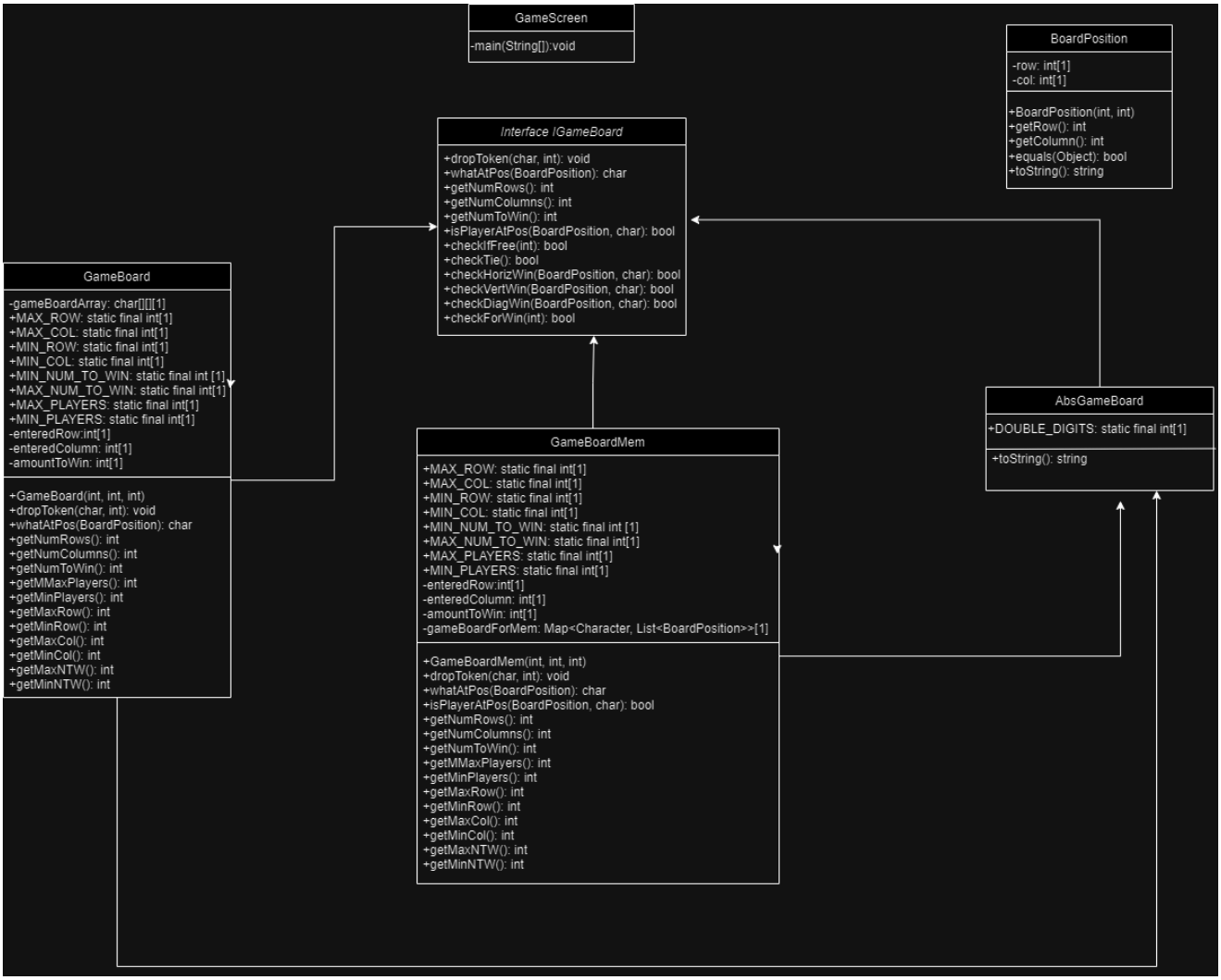
1. As a player, I need to be able to select how many rows are on the board so that I can adjust my game.
2. As a player, I need to be able to select how many columns are on the board so that I can adjust my game.
3. As a player, I need to be able to select how many pieces in a row to win so that I can adjust how I win.
4. As a player, I need to be able to select how many players there are so that I can select the correct number of tokens.
5. As a player, I need to be able to select my own unique character so I can know which token is mine.
6. As a player, I need to be able to select a fast or memory-efficient game so that I can adjust my play based on my memory needs.
7. As a player, if I choose to play again, I should be able to adjust the board sizing, number of pieces, token characters, and the efficiency of the game again so that I can play a game with a different setup.
8. As a player, I need to be able to select a column so that I can place my game piece.
9. As a player, I need to view the Extended Connect X board so that I can see what column I want to put my piece in.
10. As a player, I need to be able to see where all the pieces have been put previously so that I can decide where to put my piece.
11. As a player, I need to view the output of when a player wins to know who wins.
12. As a player, I need to know when it is my turn so that I can place my piece.
13. As a player, I need to know that my piece has been placed so that I can validate where my pieces are.
14. As a player, I need to know if where I placed my piece is a valid location so that I can place a piece again if the location is not valid.
15. As a player, I need the ability to play again so that I can play multiple games.
16. As a player, I need to be able to see where my opponent placed their piece so that I can decide where I want to put my piece.
17. As a player, I need to be able to see when a column is full so that I know I cannot put more pieces in it.
18. As a player, I need to be able to know what character I am so that I know which pieces are mine.
19. As a player, I need to be able to see how many rows there are so that I can make a strategy to win.
20. As a player, I need to be able to see how many columns there are so that I can make a strategy to win.

21. As a player, I need to know when it is the other player's turn so that I do not try to go during their turn.
22. As a player, I need to be able to see the board during the other player's turn so that I can strategize while they play.
23. As a player, I need to be able to see when there are no spaces on the board so that I know the game is over and ends in a tie.
24. As a player, I need a know that when I have placed the number of tokens set to win horizontally, I have won the game so that I know that the game has finished
25. As a player, I need a know that when I have placed the number of tokens set to win vertically, I have won the game so that I know that the game has finished
26. As a player, I need to know that when I have placed the number of tokens set to win diagonally, I have won the game so that I know that the game has finished

Non-Functional Requirements

1. The game must have a determined ending, whether it be a tie, win, or loss.
2. The game must always have (0,0) as the bottom left of the board
3. The game must be playable through the command prompt.
4. The game must ensure that each player's pieces cannot be overridden by the other player the following turn.
5. The game must be written in Java.
6. The game must be able to differentiate between the players.
7. The game must be able to differentiate between columns and rows.
8. The game must ensure that the column selected is one that exists.
9. The game must be able to take in numerical input for placing the pieces.
10. The game must be able to take in single characters for whether or not to play again
11. The game must be able to accept all English letters as valid tokens.
12. The game must be able to change the number of players when starting a new game
13. The game must have a key associated with each player token in the memory-efficient game mode.
14. The makefile must compile the files upon the call of make.
15. The makefile must run the files upon the call of make run.
16. The makefile must clean the files upon the call of make clean.
17. The game must verify that the user has entered a column number between 3 and 100.
18. The game must verify that the user has entered a row number between 3 and 100.
19. The game must verify that the user has entered a win number between 3 and 25.
20. The game must verify that the user has entered a player number between 2 and 10.
21. The game must verify that the user picks a token that has not already been chosen.
22. The game must verify that the user inputs a valid character when choosing the type of game.
23. The game must verify that the number to win is not larger than the number of columns or rows.
24. The game must convert all tokens to uppercase for comparisons.
25. The game must be able to verify win and tie conditions.
26. The game must be able to take turns between players.
27. The game must verify the character entered for playing again is valid.

UML Diagram


GameScreen
-main(String[]):void
BoardPosition
-row: int[1]
-col: int[1]
+BoardPosition(int, int)
+getRow(): int
+getColumn(): int
+equals(Object): bool
+toString(): string
Interface IGameBoard
+dropToken(char, int): void
+whatAtPos(BoardPosition): char
+getNumRows(): int
+getNumColumns(): int
+getNumToWin(): int
+isPlayerAtPos(BoardPosition, char): bool
+checkIfFree(int): bool
+checkTie(): bool
+checkHorizWin(BoardPosition, char): bool
+checkVertWin(BoardPosition, char): bool
+checkDiagWin(BoardPosition, char): bool
+checkForWin(int): bool
GameBoard
-gameBoardArray: char[][][1]
+MAX_ROW: static final int[1]
+MAX_COL: static final int[1]
+MIN_ROW: static final int[1]
+MIN_COL: static final int[1]
+MIN_NUM_TO_WIN: static final int [1]
+MAX_NUM_TO_WIN: static final int[1]
+MAX_PLAYERS: static final int[1]
+MIN_PLAYERS: static final int[1]
-enteredRow:int[1]
-enteredColumn: int[1]
-amountToWin: int[1]
+GameBoard(int, int, int)
+dropToken(char, int): void
+whatAtPos(BoardPosition): char
+getNumRows(): int
+getNumColumns(): int
+getNumToWin(): int
+getMMaxPlayers(): int
+getMinPlayers(): int
+getMaxRow(): int
+getMinRow(): int
+getMaxCol(): int
+getMinCol(): int
+getMaxNTW(): int
+getMinNTW(): int
AbsGameBoard
+DOUBLE_DIGITS: static final int[1]
+toString(): string
GameBoardMem
+MAX_ROW: static final int[1]
+MAX_COL: static final int[1]
+MIN_ROW: static final int[1]
+MIN_COL: static final int[1]
+MIN_NUM_TO_WIN: static final int [1]
+MAX_NUM_TO_WIN: static final int[1]
+MAX_PLAYERS: static final int[1]
+MIN_PLAYERS: static final int[1]
-enteredRow:int[1]
-enteredColumn: int[1]
-amountToWin: int[1]
-gameBoardForMem: Map<Character, List<BoardPosition>>[1]
+GameBoardMem(int, int, int)
+dropToken(char, int): void
+whatAtPos(BoardPosition): char
+isPlayerAtPos(BoardPosition, char): bool
+getNumRows(): int
+getNumColumns(): int
+getNumToWin(): int
+getMMaxPlayers(): int
+getMinPlayers(): int
+getMaxRow(): int
+getMinRow(): int
+getMaxCol(): int
+getMinCol(): int
+getMaxNTW(): int
+getMinNTW(): int