CPSC 2150 Project 4 Report

Evan Cox
Christian Herrman
Jonathan Maenche
Ben McDonnough

Requirements Analysis

Functional Requirements:

1. As a player, I need to be able to choose the number of players so that I can play with player groups of various sizes.
2. As a player, I need to know if the number of players I chose is invalid so that I can then choose a valid number of players.
3. As a player, I need to be able to choose what character I play as so that I can pick the character that I want.
4. As a player, I need to know if the character I choose to play as is invalid so that I can choose a different valid character.
5. As a player, I need to be able to choose the number of rows in the game board so that I can change the size of the board as desired.
6. As a player, I need to know if the number of rows I chose is invalid so that I can then choose a valid number of rows.
7. As a player, I need to be able to choose the number of columns in the game board so that I can change the size of the board as desired.
8. As a player, I need to know if the number of columns I chose is invalid so that I can then choose a valid number of columns.
9. As a player, I need to be able to choose between playing a game with a fast board or a memory efficient board so that I can choose between prioritizing speed or memory efficiency during the game.
10. As a player, I need to select a column to place a token so that I can try to win the game.
11. As a player, I need to see the current board so that I can decide where to place my token.
12. As a player, I need to see the current pieces so that I can better decide where to place my token.
13. As a player, I need to know if I have placed my token in an invalid column so that I can better decide where to place my token.
14. As a player, I can select to play again so that I can play another game of ConnectX.
15. As a player, I need to know if I have won after placing my token so that I can play again or quit.
16. As a player, I need to be able to choose a new column to drop a token into after choosing an invalid column so that I can still make my turn.
17. As a player, I need to be able to win by placing the chosen number of tokens needed to win in a row horizontally so that I can win the game.
18. As a player, I need to be able to win by placing the chosen number of tokens needed to win a row vertically so that I can win the game.

19. As a player, I need to be able to win by placing the chosen number of tokens needed to win in a row diagonally so that I can win the game.
20. As a player, I need to be able to end the game by placing a token in the last empty position on the board so that the game ends in a tie.
21. As a player, I need to be able to place a token on the board after my opponent has placed their token if they did not win so that we can continue the game.

Non-Functional Requirements

1. Must be in Java. (JDK 17)
2. The program must run on Unix.
3. The program must be a command-line application.
4. The game board is of size numRows x numColumns.
5. “Player 1” always takes the first turn.
6. Position (0,0) is the location of the bottom left position on the board.

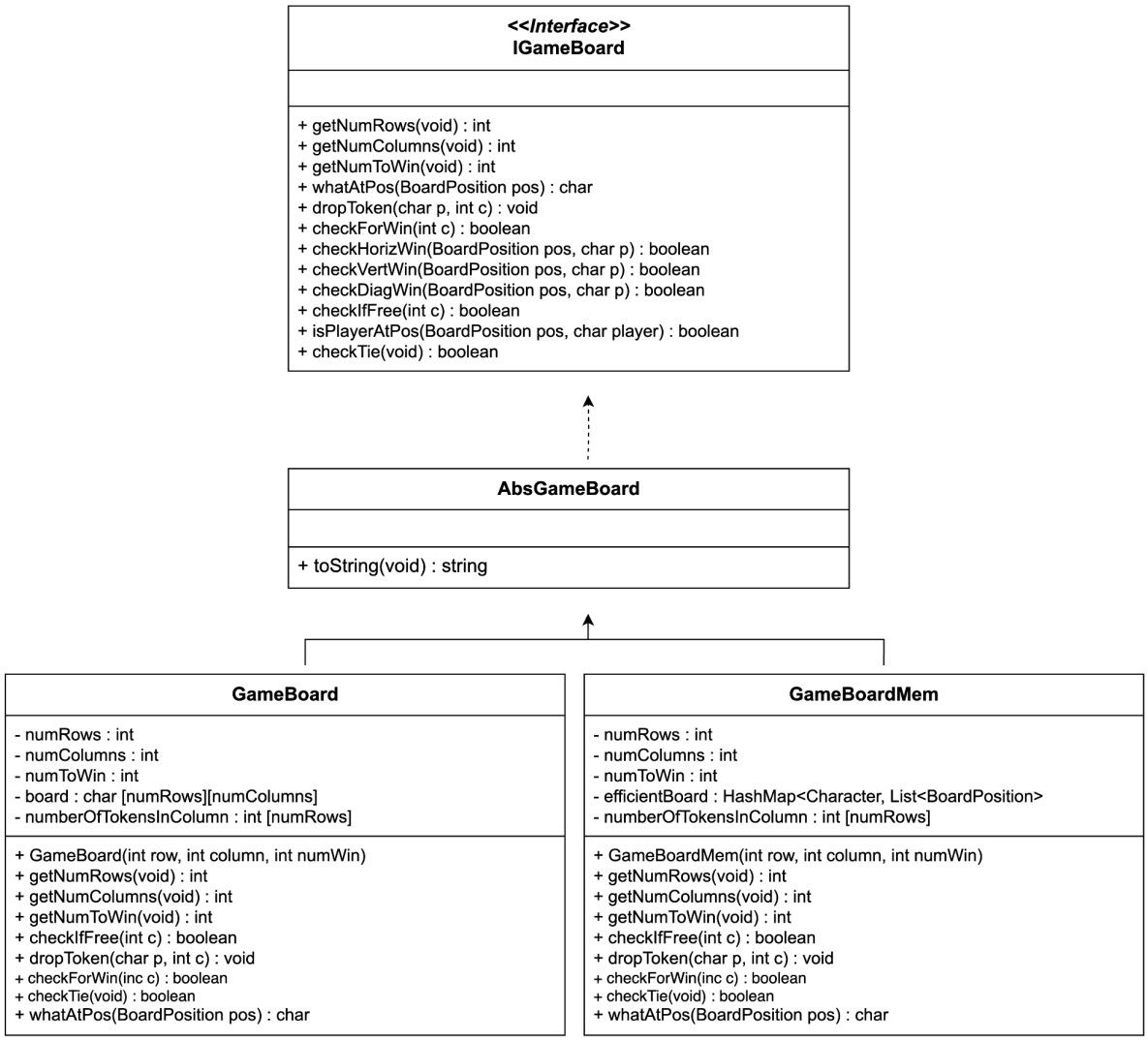
System Design

GameScreen Class Diagram:

GameScreen
+ main(args: String[]) : void - playMultipleGames(void) : void - playSingleGame(GameBoard aBoard) : void - playRoundWithPlayer(GameBoard aBoard, char p) : boolean - askPlayerForColumn(GameBoard aBoard, char p) : int

BoardPosition Class Diagram:

BoardPosition
- row : int - column : int
+ BoardPosition(int aRow, int aColumn) + getRow(void) : int + getColumn(void) : int + equals(Object obj) : boolean + toString(void) : String


<<Interface>>
IGameBoard
+ getNumRows(void) : int
+ getNumColumns(void) : int
+ getNumToWin(void) : int
+ whatAtPos(BoardPosition pos) : char
+ dropToken(char p, int c) : void
+ checkForWin(int c) : boolean
+ checkHorizWin(BoardPosition pos, char p) : boolean
+ checkVertWin(BoardPosition pos, char p) : boolean
+ checkDiagWin(BoardPosition pos, char p) : boolean
+ checkIfFree(int c) : boolean
+ isPlayerAtPos(BoardPosition pos, char player) : boolean
+ checkTie(void) : boolean
AbsGameBoard
+ toString(void) : string
GameBoard
- numRows : int
- numColumns : int
- numToWin : int
- board : char [numRows][numColumns]
- numberOfTokensInColumn : int [numRows]
+ GameBoard(int row, int column, int numWin)
+ getNumRows(void) : int
+ getNumColumns(void) : int
+ getNumToWin(void) : int
+ checkIfFree(int c) : boolean
+ dropToken(char p, int c) : void
+ checkForWin(inc c) : boolean
+ checkTie(void) : boolean
+ whatAtPos(BoardPosition pos) : char
GameBoardMem
- numRows : int
- numColumns : int
- numToWin : int
- efficientBoard : HashMap<Character, List<BoardPosition>
- numberOfTokensInColumn : int [numRows]
+ GameBoardMem(int row, int column, int numWin)
+ getNumRows(void) : int
+ getNumColumns(void) : int
+ getNumToWin(void) : int
+ checkIfFree(int c) : boolean
+ dropToken(char p, int c) : void
+ checkForWin(inc c) : boolean
+ checkTie(void) : boolean
+ whatAtPos(BoardPosition pos) : char