CPSC 2150 Project 1

Connect 4

Michael Ellis, Cooper Taylor, Ryan Chen, Adam Niemczura

Requirements Analysis

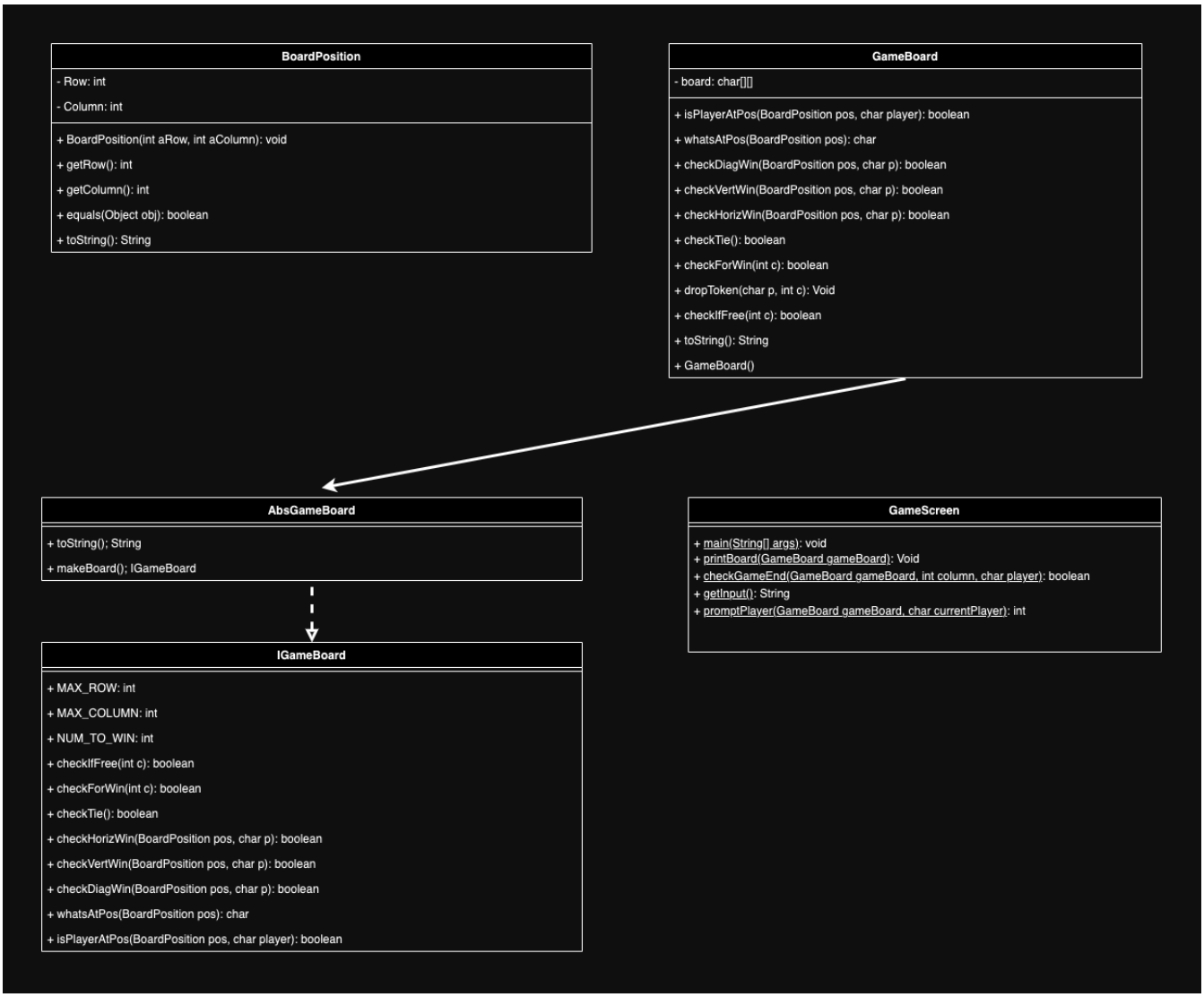
Functional Requirements:

- As a player, I need to place a marker on the board, so that I can play the game.
- As a player, I should be able to pick another column if the column I picked was full, so that I can place a piece during my turn
- As a player, I should be able to pick a column again if the one I picked was invalid, so I can place a piece on the board during my turn.
- As a player, I should be able to play until I get 5 markers in a row, so that I can win the game.
- As a player, I should be able to play until I get 5 markers in a column, so that I can win the game.
- As a player, I should be able to play until I get 5 markers diagonally, so the game ends in a win. As a player, I should be able to play the game until the board is full, so that the game ends in a tie.
- As a player, I should be able to take turns with my opponent, so that we can play against each other.
- As a player, I should know whose turn it is currently, so that I can determine who needs to place a marker.
- As a player, I should be asked to play again, so that I can keep playing with my opponent after the current game ends.
- As a player, I need the board to be printed on the screen after each turn, so that I can see the game as it is being played.

Non-Functional Requirements

- The board should have 9 rows and 7 columns
- There should be 2 players, called X and O
- A player must have 5 tokens in a row either horizontally, vertically, or diagonally to win
- Gravity should affect the tokens
- The number of tokens in a column should not exceed the number of rows
- An error message should be returned if a player tries to place a token in an invalid column.
- The program should have 2 end conditions: win or tie
- The game will start with player X
- Players should alternate placing tokens
- The game should be able to repeat once the game has ended
- Empty positions on the board should be indicated by a single space character The program should run on Linux and Windows
- The program will be a command-line application, and run in the terminal
- The program needs to be written in Java

System Design – (UML diagrams)


BoardPosition
- Row: int
- Column: int
+ BoardPosition(int aRow, int aColumn): void
+ getRow(): int
+ getColumn(): int
+ equals(Object obj): boolean
+ toString(): String
GameBoard
- board: char[][]
+ isPlayerAtPos(BoardPosition pos, char player): boolean
+ whatsAtPos(BoardPosition pos): char
+ checkDiagWin(BoardPosition pos, char p): boolean
+ checkVertWin(BoardPosition pos, char p): boolean
+ checkHorizWin(BoardPosition pos, char p): boolean
+ checkTie(): boolean
+ checkForWin(int c): boolean
+ dropToken(char p, int c): Void
+ checkIfFree(int c): boolean
+ toString(): String
+ GameBoard()
AbsGameBoard
+ toString(); String
+ makeBoard(); IGameBoard
GameScreen
+ main(String[] args): void
+ printBoard(GameBoard gameBoard): Void
+ checkGameEnd(GameBoard gameBoard, int column, char player): boolean
+ getInput(): String
+ promptPlayer(GameBoard gameBoard, char currentPlayer): int
IGameBoard
+ MAX_ROW: int
+ MAX_COLUMN: int
+ NUM_TO_WIN: int
+ checkIfFree(int c): boolean
+ checkForWin(int c): boolean
+ checkTie(): boolean
+ checkHorizWin(BoardPosition pos, char p): boolean
+ checkVertWin(BoardPosition pos, char p): boolean
+ checkDiagWin(BoardPosition pos, char p): boolean
+ whatsAtPos(BoardPosition pos): char
+ isPlayerAtPos(BoardPosition pos, char player): boolean