TEAM DOGS

Connor Burkart

Sean Farrell

Prahalad Gururajan

Parthiv Patel

Functional Requirements (17)

1. As a player, I can enter a column and place a maker.

2. As a player, I can win 5 in a row diagonally with either X's or O's.

3. As a player, I can win 5 in a row vertically with either X's or O's.

4. As a player, I can win 5 in a row horizontally with either X's or O's.

5. As a player, if I draw it will ask to play again.

6. As a player I can choose to play team O or team X.

7.) As a player, I can lose a game

8.) As a player, I can draw a game with either X's or O's.

9.) As a player, I can choose not to play again.

10. As a player, if I lose, I can play again.

11. As a player, the the position is taken, I have to chose another spot

12. As a player, I have to pick again if the column was non-existent

13. As a player, If my opponent did not win then I move again

14. As a player, I can enter a row number through 0-6.

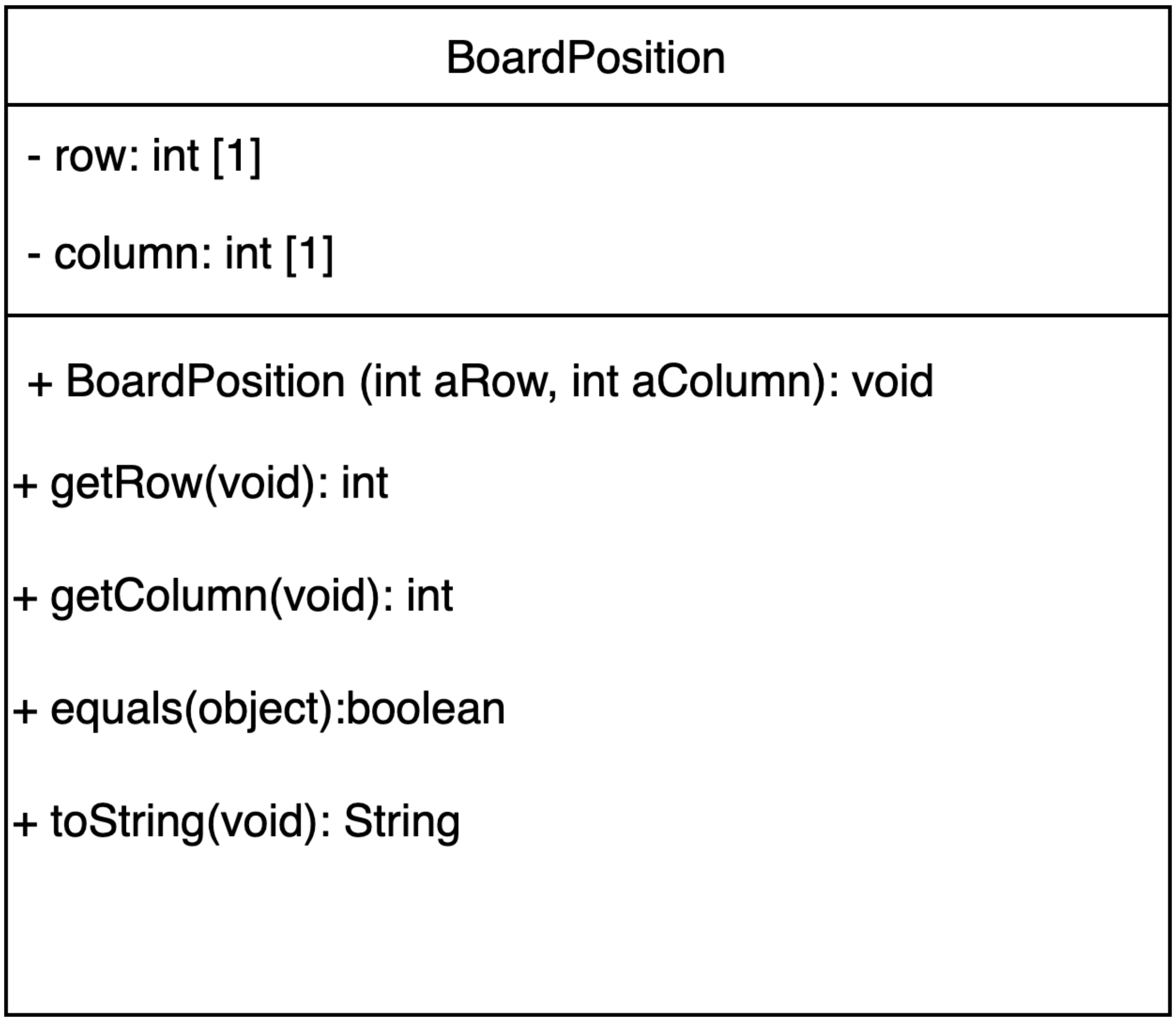
15. As a player, I can enter a Column number through 0-8.

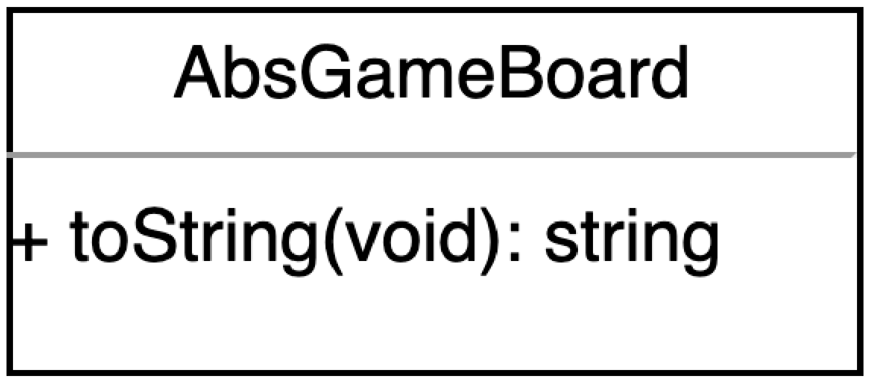
16. As a player, the the position is full, I have to chose another spot

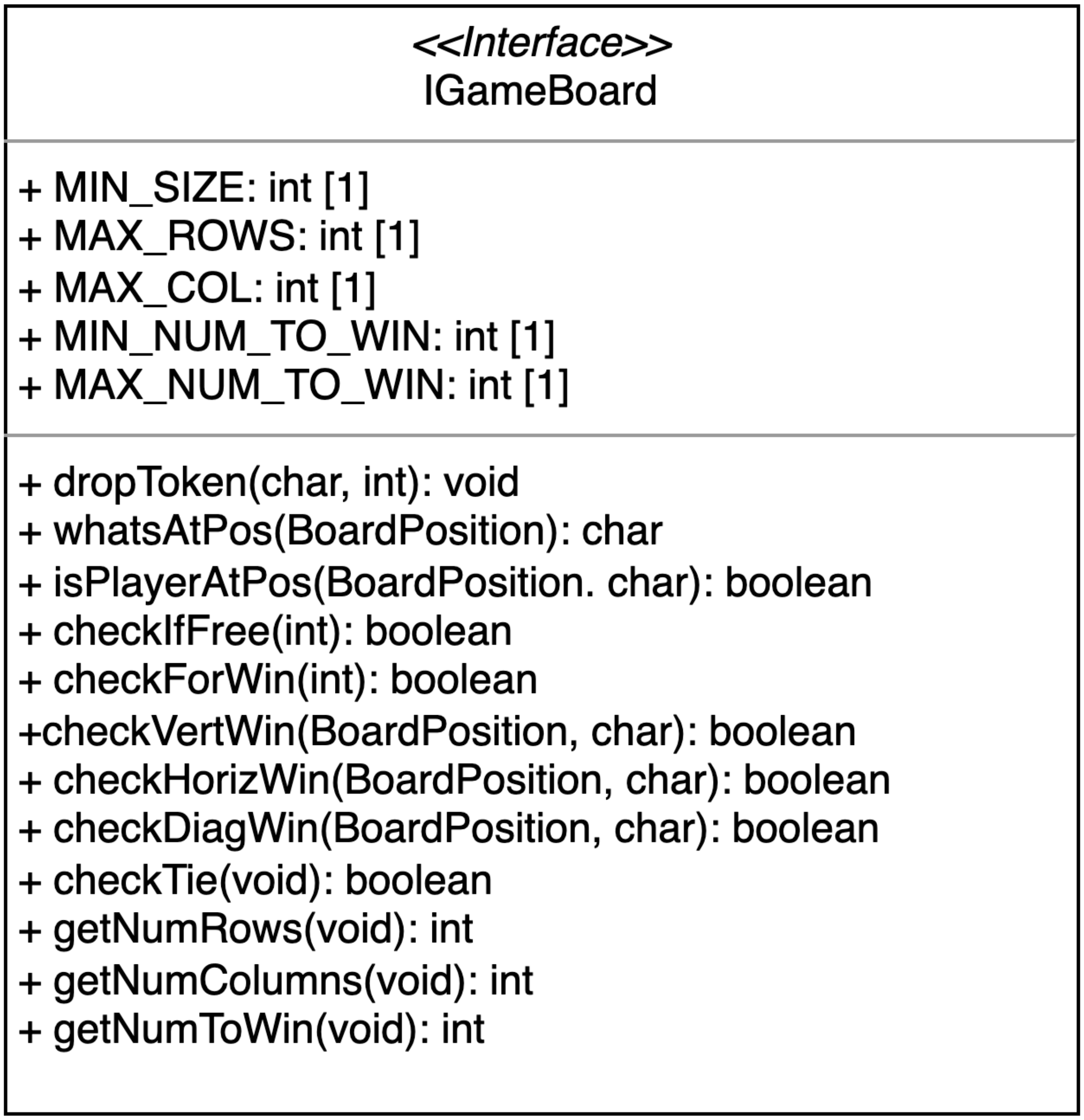
17. As a player, if my opponent did not win I have another move.

Non-Functional Requirements (5)

1. Must be able to run on a linux system.
2. The program must be able to read commands quickly.
3. The program must have enough RAM to run the program.
4. The program has to be in java.
5. The program has to be connected to the internet
6. The board size is 9X7
7. X will always go first
8. (0,0) is the bottom left corner


BoardPosition
- row: int [1]
- column: int [1]
+ BoardPosition (int aRow, int aColumn): void
+ getRow(void): int
+ getColumn(void): int
+ equals(object):boolean
+ toString(void): String

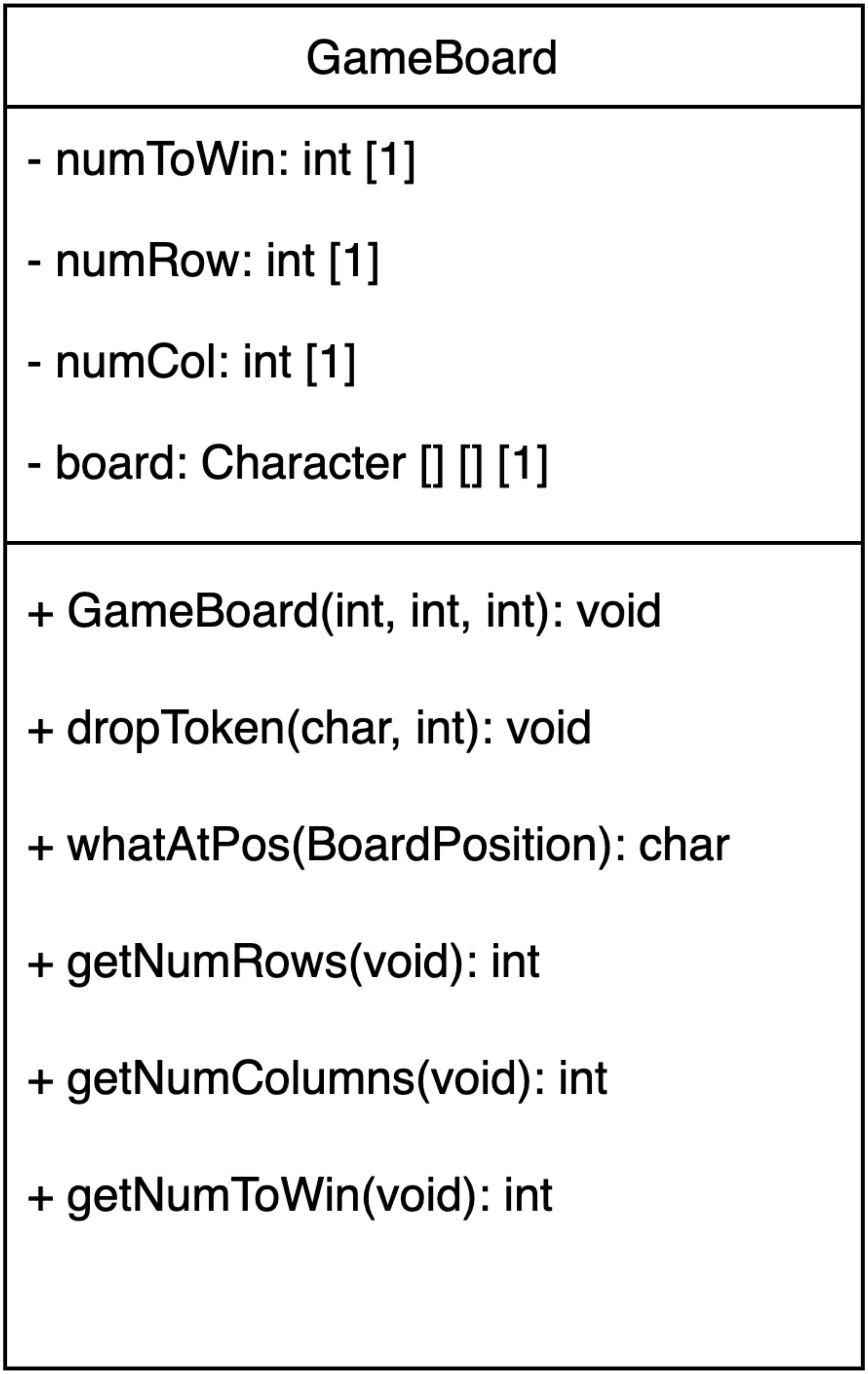

AbsGameBoard
+ toString(void): string

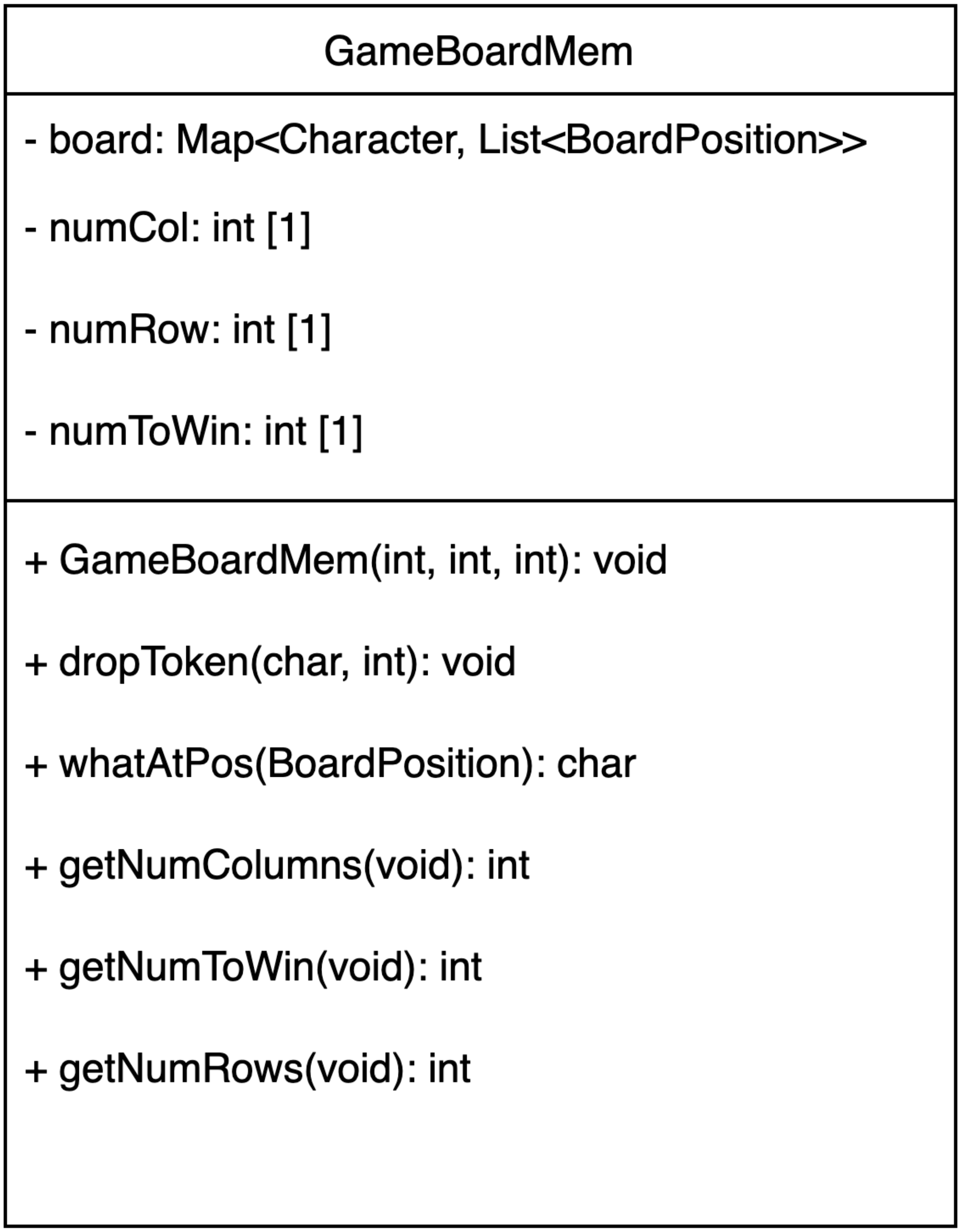


<<Interface>>
IGameBoard

+ MIN_SIZE: int [1]
+ MAX_ROWS: int [1]
+ MAX_COL: int [1]
+ MIN_NUM_TO_WIN: int [1]
+ MAX_NUM_TO_WIN: int [1]

+ dropToken(char, int): void
+ whatsAtPos(BoardPosition): char
+ isPlayerAtPos(BoardPosition. char): boolean
+ checkIfFree(int): boolean
+ checkForWin(int): boolean
+checkVertWin(BoardPosition, char): boolean
+ checkHorizWin(BoardPosition, char): boolean
+ checkDiagWin(BoardPosition, char): boolean
+ checkTie(void): boolean
+ getNumRows(void): int
+ getNumColumns(void): int
+ getNumToWin(void): int

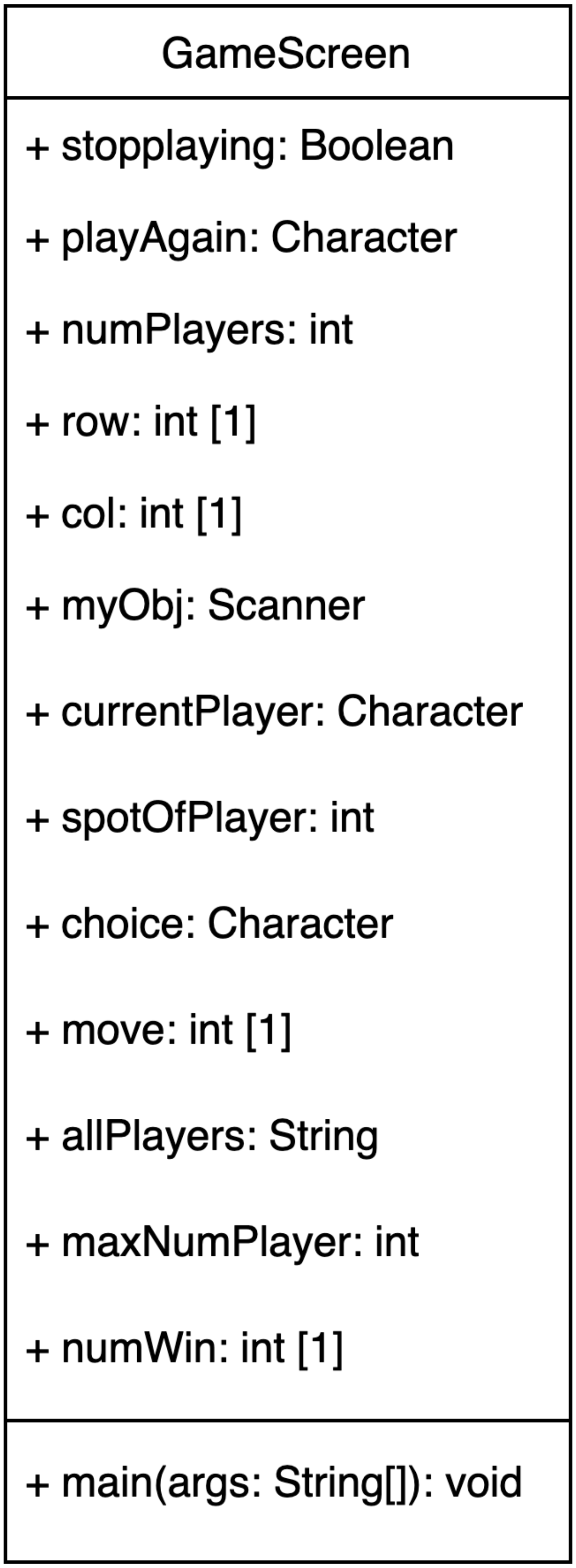

GameBoard
- numToWin: int [1]
- numRow: int [1]
- numCol: int [1]
- board: Character [] [] [1]
+ GameBoard(int, int, int): void
+ dropToken(char, int): void
+ whatAtPos(BoardPosition): char
+ getNumRows(void): int
+ getNumColumns(void): int
+ getNumToWin(void): int



GameBoardMem

- board: Map<Character, List<BoardPosition>>
- numCol: int [1]
- numRow: int [1]
- numToWin: int [1]

+ GameBoardMem(int, int, int): void
+ dropToken(char, int): void
+ whatAtPos(BoardPosition): char
+ getNumColumns(void): int
+ getNumToWin(void): int
+ getNumRows(void): int


GameScreen
+ stopplaying: Boolean
+ playAgain: Character
+ numPlayers: int
+ row: int [1]
+ col: int [1]
+ myObj: Scanner
+ currentPlayer: Character
+ spotOfPlayer: int
+ choice: Character
+ move: int [1]
+ allPlayers: String
+ maxNumPlayer: int
+ numWin: int [1]
+ main(args: String[]): void