CPSC 2150 Project Report

Jason Russo
Joseph Becker
Evan Schwartz

Requirements Analysis

Functional Requirements:

1. As a player, I need to be able to choose a column to drop a chip so I can take a turn.
2. As a player, I need to know which columns are full, so I do not overload one.
3. As a player, I need to know whose turn it is, so I do not act out of turn.
4. As a player, I need to know the size of the board, so I do not select a column out of the bounds.
5. As a player, I need to know when the game is over, so I know when to stop making selections.
6. As a player, I need to know who won the game, so I can track wins over other players.
7. As a player, I need to have the option to play again so multiple games can be played.
8. As a player, I need to know if the game resulted in a tie so I can quit making moves and track my record.
9. As a player, I need to know if I am X or O
10. As a player, I need to be informed if my choice of move is invalid
11. As a player, I can see the game board after each turn so that I can stay informed on the progress of the game.
12. As a player, I need to be presented with a fresh board if I choose to play again, so I can start a game from scratch.
13. As a player, I need to be able to be informed of the rules of the game if I do not know them
14. As a player, I need to know which number corresponds with each row
15. As a player, I need to know which number corresponds with each column
16. As a player, I need to be informed of the controls for the game
17. As a player, I need to know if my last placed token completed the 5 same tokens in a row horizontally to win the game.
18. As a player, I need to know if my last placed token completed the 5 same tokens in a row vertically to win the game.
19. As a player, I need to know if my last placed token completed the 5 same tokens in a row diagonally to win the game.

Non-Functional Requirements

1. The game must run on Unix
2. The game must run on the command line
3. The program must be written in Java
4. The board size is 9x7
5. X always goes first

6. (0,0) is the bottom left position of the board

System Design

GameBoard:

Class diagram

GameBoard
- board: char[][]
+ GameBoard(): void + checkIfFree(int c): boolean + 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 + whatsAtPos(BoardPosition pos): char + isPlayerAtPos(BoardPosition pos, char player): boolean

GameScreen:

Class diagram

GameScreen
+ main() : void

BoardPosition:

Class diagram

BoardPosition
- Row: int - Column: int
+ BoardPosition(aRow: int, aColumn: int) + getRow(): int + getColumn(): int + equals(obj: Object): bool + toString(): string

<<interface>> **IGameBoard**
+ NUM_ROWS: int + NUM_COLS: int + NUM_TO_WIN: int
+ getNumRows(): int + getNumColumns(): int + getNumToWin(): int + checkIfFree(int c): boolean + dropToken(char p, int c): void + 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

AbsGameBoard
+ toString(): String