**Design Document - ICS3U Culminating**

**Overview**

My program is connect four. It will be used as entertainment for kids or anybody who is bored or wants to play a game, or pass the time. The game will be able to print a 6x7 board, ask the users where they want to play their game pieces, and a coloured circle will appear where they enter. If a player wants to put their game piece in a column that is already full, or a number not in the range of 1-7, the player will be asked to put their piece somewhere else. The program will be able to detect 4 in a row horizontally, vertically, and diagonally, and will print who won. It can also detect when the board is full, and there are no 4 in a row, it will then print out that it’s a tie, and is also be able to replay once the game is finished.

**Diagrams**

**The program will start with an explanation of the game rules, printing a numbered, empty 6x7 board, and asking the first player (red) where they want to place their game piece:**

Game Rules:

-There are 2 players that take turns placing their game pieces, red goes first, yellow goes second

-The game is played on a 6x7 grid, the player will enter the number column on the grid they want to place their game piece

-The piece will go to the lowest empty space in the column entered

-The first player to get 4 of their pieces in a row horizontally, vertically, or diagonally wins

Red player, where do you want to place your game piece?

1 2 3 4 5 6 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Eg. Red player inputs 6**

**The board will print again, including where the red circle was placed, and ask the second player (yellow) where they want to place their game piece.**

Yellow player, where do you want to place your game piece?

1 2 3 4 5 6 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Eg. Yellow player inputs 1**

**The board will print again, and red and yellow will continue to take turns placing their pieces.**

Yellow player, where do you want to place your game piece?

1 2 3 4 5 6 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**If a player wants to place their game piece in a column that’s already full, the program will ask the player to place their game piece in another place.**

Red player, where do you want to place your game piece?

1 2 3 4 5 6 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Eg. Red player inputs 8**

That column is full, please enter a different number.

**If a player wants to place their game piece in a column that doesn’t exist (not in the range of 1-7), the program will ask the player to place their game piece in another place.**

Yellow player, where do you want to place your game piece?

1 2 3 4 5 6 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Eg. Yellow player inputs 8**

This spot is not on the grid, please enter a different number.

**The program can detect when there are 4 red or yellow in a row horizontally, vertically, and diagonally, and will print the winner. The program will then ask the user if they want to play again. If the user enters anything other than yes or no, the user will be asked to enter it again.**

1 2 3 4 5 6 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Game over, red player wins!

Do you want to play again? (yes/no).

**The program can detect when the board is full, and will ask the user if they want to play again. If the user enters anything other than yes or no, the user will be asked to enter it again.**

1 2 3 4 5 6 7

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Game over, it’s a tie!

Do you want to play again? (yes/no)

**Top - Down Design**

**public static void main(String[] args){ -** The main method calls all of the other methods and prints questions and statements to the screen for the user to answer/see. This method will print which player wins, if the board is full and the players tied, and will ask the user if they want to play again. The main method also declares a 2D array which will store the 1 and 2 (red and yellow) values.

**public static int playerTurn(int player){ -** The playerTurn method will get the value 1 or 2, 1 being red, 2 being yellow, and the method will ask whichever player who’s turn it is where they want to place their game piece. It will return the column number that the player wants to place their game piece.

**public static void printGrid(int[] grid){ -** The printGrid will print the grid according to the given array of numbers that represent the colours. It will print the outline as well as coloured circles in the spots previously entered by the players. This method doesn’t return any value.

**public static int horizontalWin(int[] grid){ -** The horizontalWin method checks if there are any 4 of the same colour (1 or 2 in the array) in a row horizontally. It will return 1 if there are 4 reds in a row horizontally, 2 if there are 4 yellows in a row horizontally, or 0 if there are no 4 in a row horizontally.

**public static int verticalWin(int[] grid){ -** The verticalWin method checks if there are any 4 of the same colour (1 or 2 in the array) in a row horizontally. It will return 1 if there are 4 in reds in a row vertically, 2 if there are 4 yellows in a row vertically, or 0 if there are no 4 in a row vertically.

**public static int diagonalWin(int[] grid){ -** The diagonalWin method checks if there are 4 of the same colour (1 or 2 in the array) in a row diagonally top left to bottom right or bottom left to top right. It will return 1 if there are 4 reds in a row diagonally, 2 if there are 4 yellows diagonally, or 0 if there are no 4 in a row diagonally.

**public static boolean fullColumn(int[] grid, int column){ -** The fullColumn method will see if the column that the player wants to put their game piece in is full or not. If the column is full, the method will return false, if the column is not full yet, the method will return true.

**Public static boolean correctColumn(){ -** The rightColumn will return true if the player entered a number in the range (inclusive) of 1-7, since there are only 7 columns. It will return false if the player inputs any number that doesn’t have a corresponding column (eg. 0, 8). This method doesn’t need parameters since there are always 7 columns, and no variable will change that.

**public static boolean fullBoard(int[] grid){ -** The fullBoard method will go through all of the spots in the array, and check if there is something in all of the spots. If the board is full, the method will return false, if the board is not full yet, the method will return true.

**public static void clearBoard(int[] grid){ -** The clearBoard method clears the board by taking the 1 and 2 (red and yellow) values out of all of the spots in the array. This method doesn’t require parameters, since it only needs to go through the array, regardless of what values are already stored in it. It also doesn’t return anything, since the values in the array will change, and when the grid is printed next, it will be empty.