Technical Document (Connect Four Game)

Nina Di Cara

Time Spent on Project

I had one week to complete due to not having access to the task originally. The project took 5 days with around 4 hours a day spent working.

Design

MVC Design: Before deciding on whether to implement the GUI or to use ASCII characters I first designed the Model View Controller. I considered the objects involved in a game of Connect Four, namely the playing grid, the pieces, the players and then the ongoing game itself. From these I decided on two main classes: Game and Grid. Grid was to deal with methods which interacted with the playing grid and Game was to deal with movements made and other game related activities, including checking whether the player had won.

hasWon() : I spent a lot of my coding time trying to get hasWon() to correctly identify a winner. I chose for loops to find horizontal and vertical winners by searching outwards from the piece last played – this was an attempt to save some CPU time instead of searching the whole board. Horizontal searches either side, keeping a count on each which returns true if the count reaches the winning number of pieces. The same for vertical except it was not necessary to search upwards since moves could have only been made below if looking at the most recent piece.
My diagonal methods were originally in two nested for loops, however I had trouble getting these to work so I switched to a while loop which did sometimes provide the right result. The counter principle was employed again to enable a separate search in both directions while keeping count of the number of pieces in a row.

Extra Functionality: I wanted to include some of the extra functionality so my game can be played with a custom number of players (between 2 and 6), custom rows and columns and a custom winning number of pieces in a row. I have limited the number of players to 6 purely because I wanted each player to have a different colour piece and there are only so many distinctive colours available.

GUI Design: The next big design decision I made was regarding how to implement my GUI. I knew I wanted to use a GUI for the grid and not just ASCII characters as I wanted the game to look and feel professional. I had the option of creating a lot of JButtons with their own Action Listeners but since I wanted the user to be able to choose how many rows and columns to play and I didn't know how I would get JButtons and Action Listeners to generate within a loop I needed another option. The far better option seemed to be painting the grid by extending JPanel, which could be done inside a loop so as to enable as many rows and columns as the user wanted, and then getting the coordinates of a click with the Mouse Listener. These coordinates could then be used to figure out which column was clicked on and call makeMove from the Game class.

Data Structures

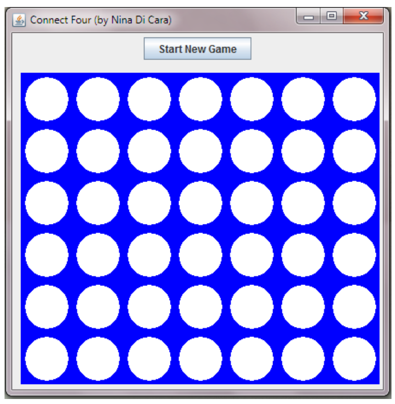
To store the playing grid I chose a 2D array of integers. Originally I wanted to use a character array and have players as A, B, C... but then realised that if I used numbers it would be a lot simpler to increment the player each move and to make calculations involving the grid. I could then still set a colour to each number in the GUI to represent which player has dropped which piece.

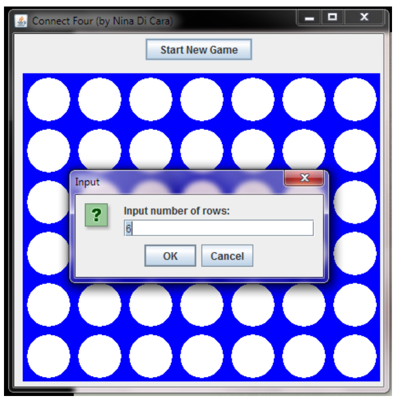
Problems and Limitations

These are some of the problems or limitations that I have noticed in my project. I found myself pushed for time to complete the project in the week whilst revising for exams so many of these issues are ones which I am aware of and I feel I could have resolved had I more time to spare.

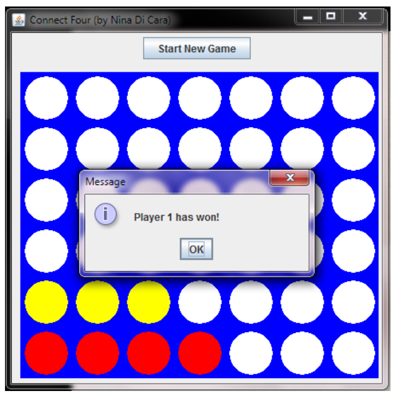
- I did not manage to get my diagonal searches completed in time and this is a clear problem with the game. I feel these could be made more reliable with some attention.
- In C4ButtonPanel, catching the NumberFormatException exits the Button Listener method, if I had more time I would develop this and maybe use a different form of error checking to stop this happening.
- In C4ButtonPanel, pressing “cancel” displays the dialogue box prompting for an integer input. This is also something that could be fixed with more time.
- I would have liked the programme to display an error if the number of columns/rows or central diagonal is less than the number of pieces in a winning line.
- I originally intended to have a counter above the game which showed the current player number and what colour piece they were using. This is something I would implement given more time.
- Instead of having the user inputting a number of players and having to error check I considered having Radio Buttons, which may have given a more sophisticated feel. I could have also had drop down menus for the other input options to make the game feel more professional.
- An AI element of game play against the computer would have been interesting to try but in the timeframe and given my lack of experience in writing code for this purpose I chose not to implement this feature.
- I would have liked the JFrame to resize based on the size of the JPanels but was not certain how to implement this. Also, if maximised I would have liked the game grid to be central on the screen..

Screen Shots of the Game





Input number of rows:
6



Player 1 has won!