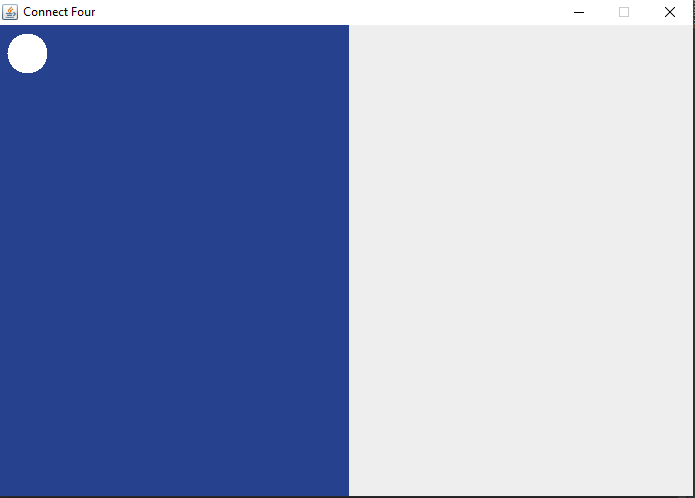
**Connect 4 Mini Project**

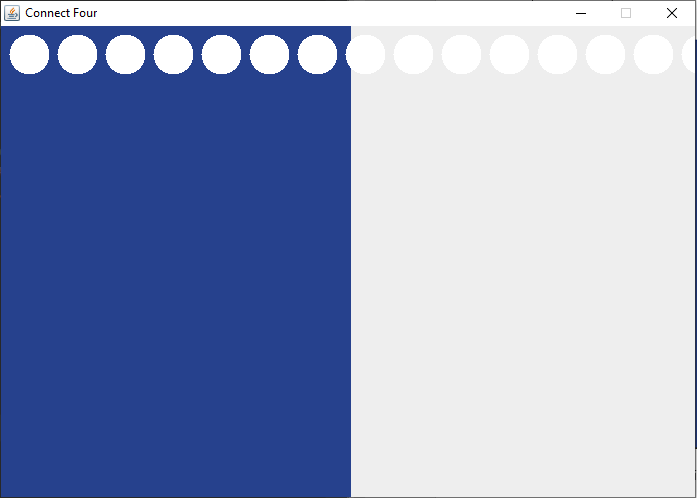
This is my attempt at designing a connect 4 game 😊

My initial job was creating a Frame. I set out by creating a simple JFrame object and set values for title size and making it so that the frame was not expandable. I then created a shapes class that extended JPanel to draw my frame on to my screen. This shapes class contained a single argument constructor that took in the Dimension argument for the size of the Frame (The dimension class was something I discovered and decided to add it to the project. Alternatively, I could have hardcoded it in the frame size) I created a 2D grid Array of the Color object to allow me to paint my rows and columns. The grid array was then instantiated in the single argument constructor by creating 2 for loops with the outer most for loop controlling the Y-Axis of the Grid and the inner Loop controlling the X-Axis and set the Colour of each cell of the grid object (I set it to white to begin with). I then created a paintComponent object this was something I researched so that I could display my grid, my holes for my counters and my counters onto my JFrame window. I began by painting a rectangle canvas so that I could create my frame for my actual connect 4 board (I chose a classic blue here 😉). I then created another double for loop here so that I could paint the grid on the screen.

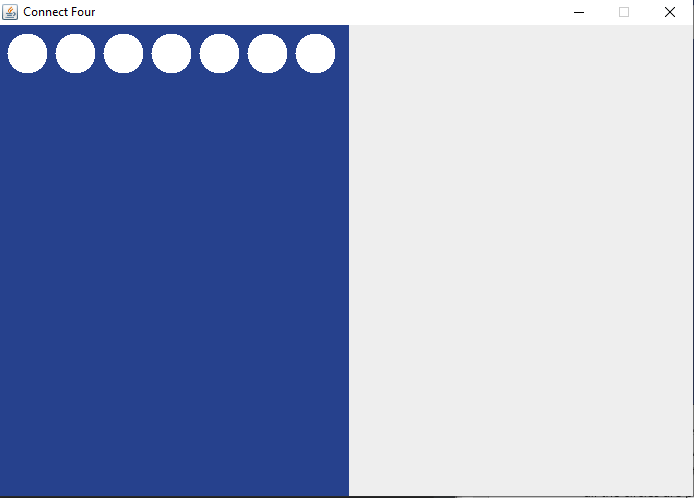
When painting the grid on the screen I had to Increment the X and Y-Axis values so that the white(Blank) circles would paint across the screen. I initialised these values at 8 each so that they were separated from the border. When painting the values without incrementing the values all the circles painted on top of each other and appeared like this(all 42 circles are on top of each other):



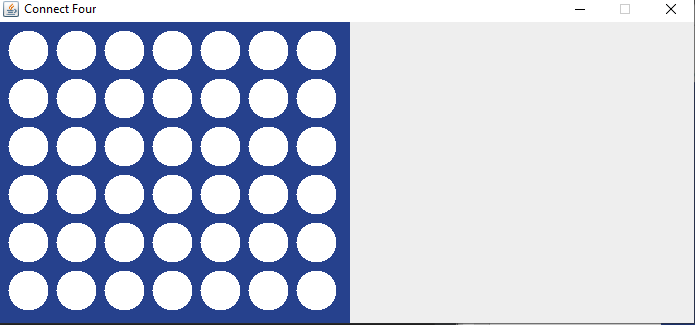
I then incremented my starting x position and added on the circle width and a gap (for style) and I got this:



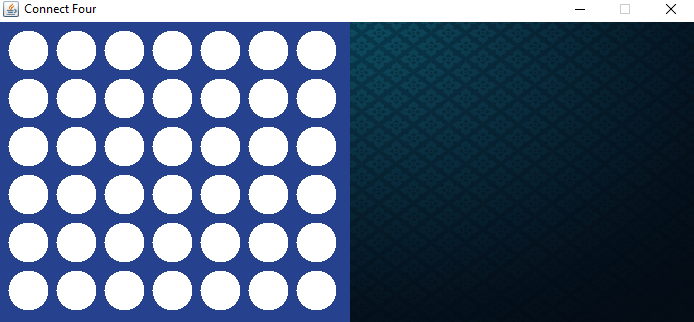
This created all the Circles in one row so ounce the first row had painted the circles (which is circle 7 in the picture) the row didn’t reset so the next circle was painted along side circle 7 and this makes it run off the screen. On the Y loop if we reset the XAxis value back to 8(this was my starting position) then all the circles are painted on top of each other again, but they still paint across the row. This is what this looked like:



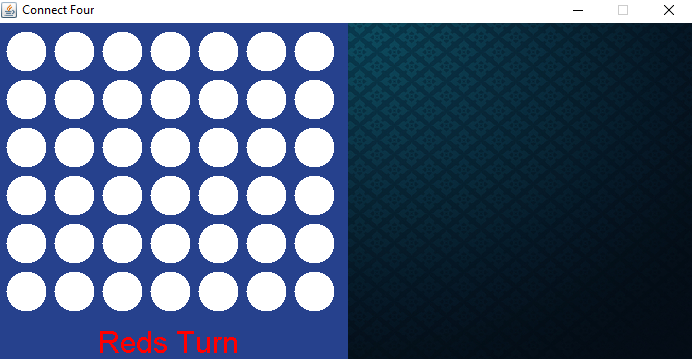
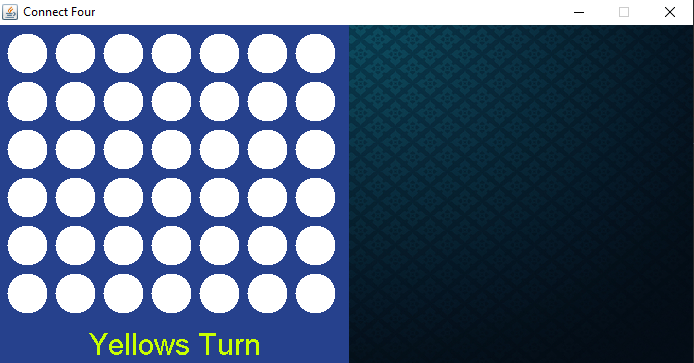
So now I have 1 row still, but I have 7 columns created. Well technically I have 6 rows they are just all painted on top of each other. So if I added the starting Y Axis to the circles width and the gap then the Ovals starting y position will change so that the rows will print vertically. So the grid then appears like this:



Hooray 😊 so I have a grid on the screen. With the grid created I decided to add a Background Image (For the crack) to the whole JFrame window. (I looked up a video on how to add pictures to Java.) I simply created a JLabel for the background image and the ImageIcon class is used to retrieve the image in the same package (I could have created an images folder which could contain the images and simply called /images/backgroundImage.jpg but I taught this was not necessary as I might not need many more images)(**EDIT** this changed I did create an image folder and I had to repaint the background each time so it is no longer a JLabel) so this is what I had at this stage **(16/11/20):**



I am using the right part of the screen as a sort of menu to display information about the game so it will be my sort of scoreboard. I started this by displaying the name of whose turn it was and displaying it in the colour of that player.



Next I used the MouseListner. Before I began this I adjusted the size of the circles from 40 to 50 (Just to make it more aesthetically pleasing) I will admit the MouseListner took me some time to implement but I am happy with the result. I implemented this in the DrawShapes class (**EDIT** gameboard class) as I can use this to redraw shapes when the mouse is pressed. I determined that I had to figure out a way of calculating each circle in the grid to do this I first got the X axis and Y axis using the e.getX(); and e.getY(); this gave me the both the x axis and y axis in pixels. So I took the very top left cell as an example on how I would determine each cell.

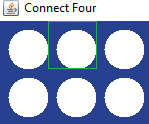
So each circle was 50 pixels wide and each cell also includes an 8 pixel gap both vertically and horizontally. So I took it that each block of circles and gaps was 58x58



A cell block included the left gap of 8 and the top gap of 8 (so sadly if you click in these areas it will count as putting in a counter in that slot)

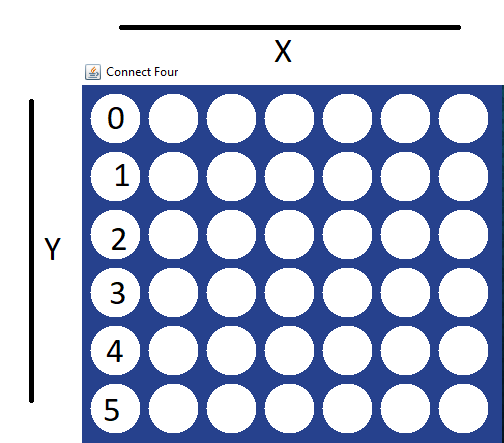
Starting on the x axis at 0 the mouse moves 58 pixels to the right then on the 58th pixel the mouse will have moved into the second cell. So if I take the x position of the mouse and divided that by 58 the total width of a circle plus the gap to the left then on the first cell I will get a result of 0 as no number on the x axis of the mouse position within that cell will result in a number greater than 58 (this 0 is 0 on my grid array) as I am using Integers there will be no remainder. When the mouse goes into the second cell the number on the mouse press will be a number between 58-115 so when the mouse is clicked the number will be divided into the circle width plus 8 (e.g (65/50+8) as these are Integers this will result in a 1) this will be [1] on the “row” for my 2D grid Array. The same principle applies for the y-axis so from top down it is 58 pixels so the mouse starts in the top corner at 0 pixels as it moves down the screen the number increases when it goes over 58 it has reached the second cell when it reaches 116 it is in the third cell…..etc.

**2nd Cell**



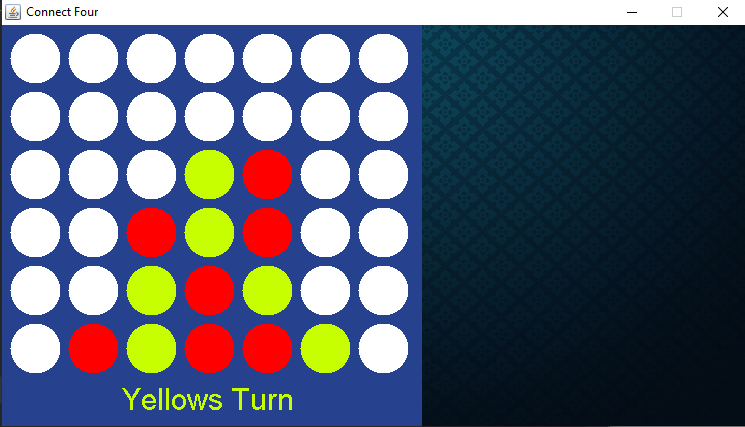
After all that it was just simply set up a simple validation that determines if it is red or yellow playing and change the grid to the colour of the chosen player. (That was the easy part 😊) **(18/11/20)**

I started off my next session creating a methods class so I could keep the code tidy 😊 my next goal was to make it so that the now established red and yellow circles would drop down a column ONLY if the occupied colour was not red or yellow if the occupied colour was red or yellow then the colours would be placed one square above that. Also if the column was less than zero(because the top most y position is zero) than it is not on the grid and would display an error message.

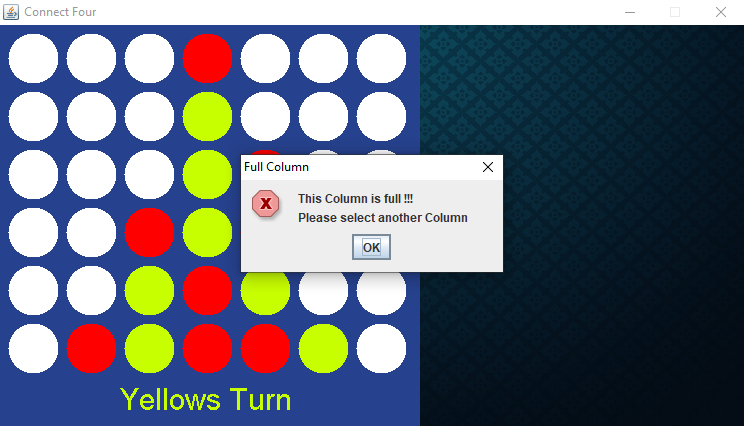


So on the Y axis the array will go from 0 to 5. If the colour at 0 is either red or Yellow (Realised this afterwards I could have also said if the colour is white to validate my positions. 6 of 1 and a half a dozen of the other 😊) then throw an error message say the column is full if not index++. If YPositon at 1 is either red or yellow then return yPosition minus one else Index ++. At the bottom row the loop will finish and return what position the counter should go in .I will need to take in the XPosition as an argument in this Method so that I know what column to drop a counter into. I will also be taking the rows as an argument (I could technically hardcode in 6 because I know there are 6 rows but I won’t encase I change it in the future, not really going to happen but still 😊) and I will also be taking in the Gird as an argument. I need the grid so I can check which colours are in the grid position currently. I created this all in a checkForFreeSpace method which returned integer values depending on the position that a counter was detected.

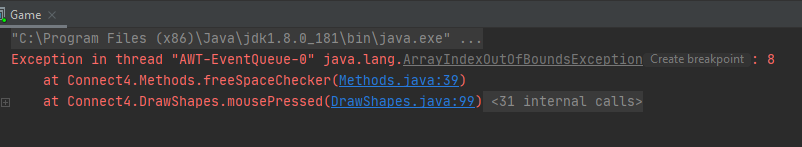
This is difficult to demonstrate but this was my end result:



You will have to take my word for it but when you select a column the counter is placed in the bottom most available spot in that column 😊 that’s a huge achievement because now without implementing checking validation you could still technically have a game of connect 4 😊 if you reach the top of your y positioned row and it contains either red or yellow it will display an error message stating that the column is full and to select another column. (**20/11/20**)

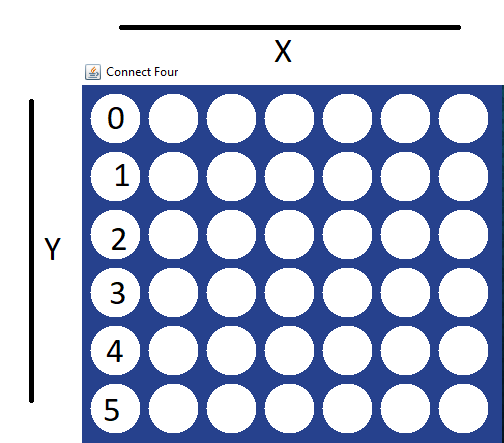


At this point I decided to implement input validation code to prevent a click away from the grid. As it stands if you click anywhere outside of the 2D grid array an error message will appear.

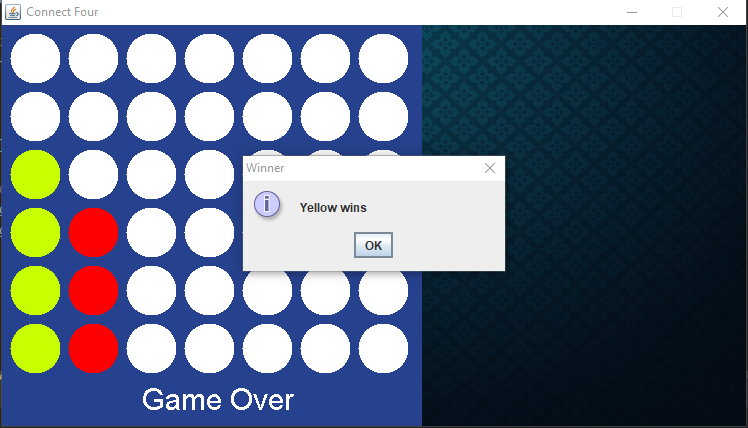


This is because the xAxis array extends beyond the grid created. Now this does absolutely nothing to the game. The game still functions as normal however this is just not good code to allow errors like that, so to prevent this I will create one simple IF statement which will check if the distance of xAxis and the distance of the yAxis is less than the size of the grid (so as long as it is within these limits the game will allow to paint on it). This solved this issue so now the clicks only apply to the game board 😊

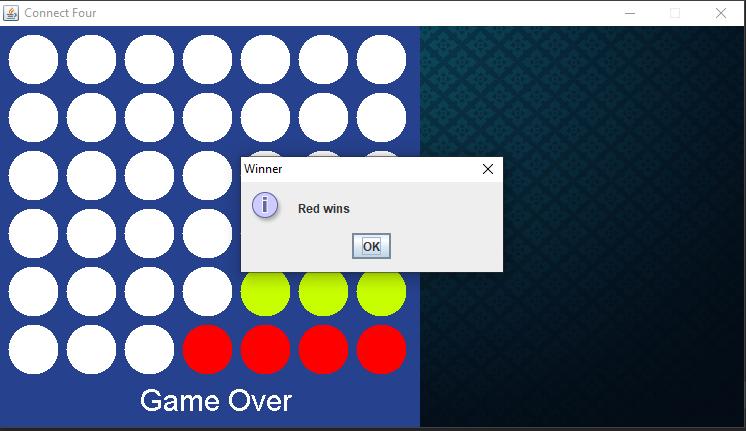
My next test will be the big one which will be checking to see if somebody has won. I will create this in the methods class. I am going to create this as a Boolean and create a counter to count if tokens in any direction add up to 4 then it will return true and declare a winner if not it will return false and check somewhere else if all checks completed and false is returned then the turn increments and the next player plays. I will begin this by checking South so this method will require a few arguments it will need the x and y positions so the counter can start checking from this position. It will need the current player colour so that it can verify the direction of the next colour is the same and of course I need the grid so that I can check the other cells. When doing the South checking algorithm, it is as simple as creating a count and checking if the counters bellow the current counter add up to give a 4. If true then the counter has created 4 in a row so it will return true and declare a winner. When checking for the next counter position I will have to increase my counter variable because the highest valued index of the array is located in the bottom half of the array.



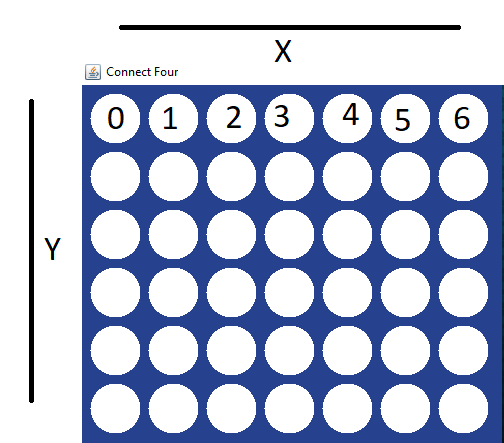
So if I placed the counter in the 2nd row of the 0 column then the next counter to check will be on the 3rd row if that is a valid counter then the count variable will increase to 2 meaning I have 2 in a row and then check the counter in the 4th row in the 0 column. This is the result:



So yellow got 4 on the 0 column and this resulted in a win. I then displayed this as a message to the screen and then froze gameplay as the game is over. I also changed the ‘player turn’ String to display game over instead of who’s turn it is. The same process applied for checking counters to the right of a placed counter. Keeping the row as a sort of constant sentinel, check if the column +1 is the same colour as the colour of the current position:

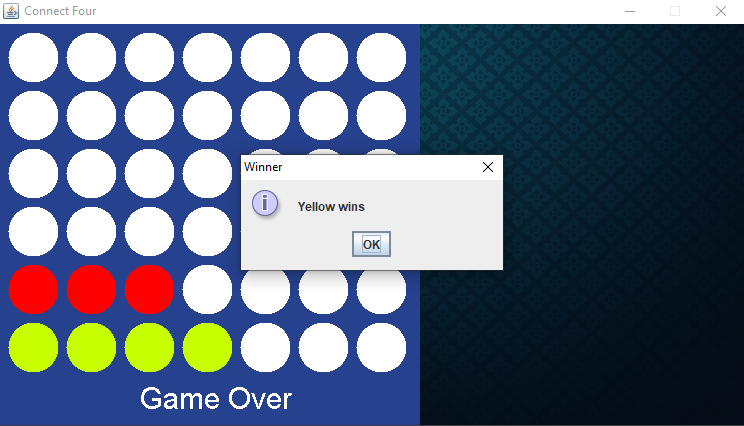


So the 3rd column was the last counter placed. It checks if the counter in the column +1 is red and since this is true the count incremented ounce the count reached 4 red won



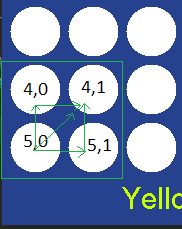
(I placed those numbers badly on paint 😊)

The same applied for checking counters to the left except this was reversed so instead of checking the column +1 I was checking the column -1:



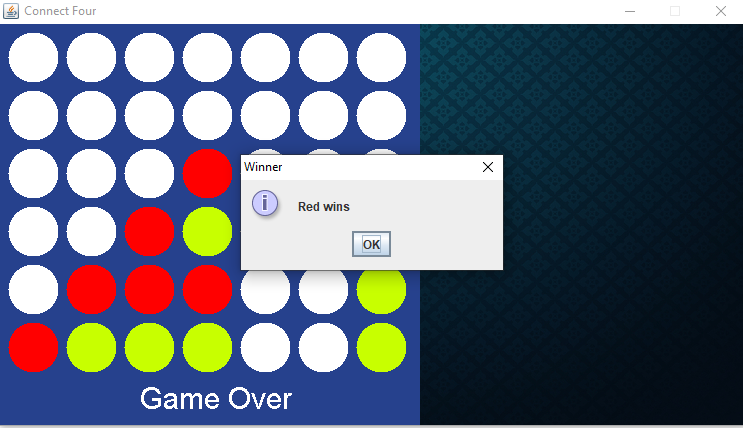
**(23/11/20)**

For the diagonal I simply check from the yAxis positive+/-1 position and change the xAxis to +/-1 one number in the same check. So for example I checked North East first so to do this I will check one column+ so it will move to the right and one row-(minus) so that it goes up the axis



This is the bottom left corner of the board and its location is row 5 column 0. My current row is 5 so to check one above it I have to take one from it and my current column is 0 so to check one to its right I must increment it

The result is this:



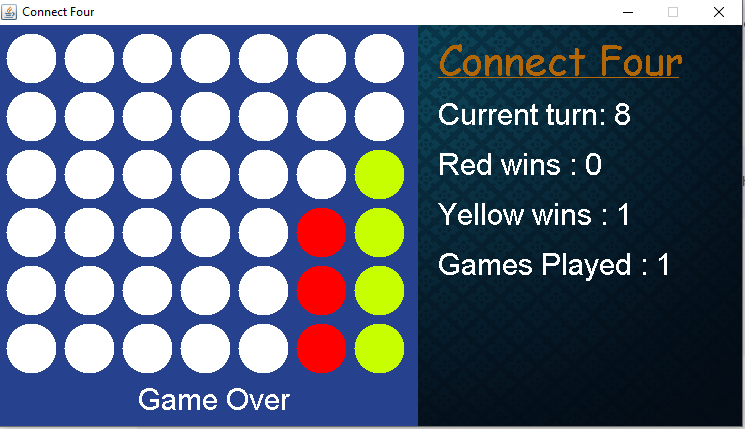
(The last counter placed is row 5, column 0)

The other diagonals are in similar fashion so to go down my value will be incrementing **(+)**, to go up my value I will decrement **(-)**, to go to my right my value will increment **(+)** and finally to go to my left my value will decrement **(-)**. I did this for the other directions and validated my worked (I did not think it was necessary to screen shot the working models of this code as they are just in diagonals similar to the image above)

At this stage my validation is complete and my game can now calculate if a player has won in any direction 😊

So there is not much left to do. I am going to implement a scoreboard to the right so the player can have information on the current game and previous winnings as well as how many games played.**(24/11/20)**

Ok so on the **25/11/20** I ran to the finish line 😊 I forgot to take screenshots along my way. I added in my score board which holds game information.



The button was the biggest implementation. The button is an image that I created in Photoshop (thanks to user interfaces last year 😉) it changes colours when clicked to red, just to let the user know it is clicked. When clicked the user is asked if they want to start a new game (encase a user miss clicked the button in the middle of a game) if user selects no(which because it’s a yes no box it returns a 1) nothing happens if yes (returns a 0) then the game will set gameOver variable to false meaning the board can be clicked. It then randomly assigns the players and proceeds to clean the board by placing all white circles on it again.

I then added some coin sounds, a clap for if you win and background music and that’s it finished 😊

**(25/11/20)**



**When you click new game**

