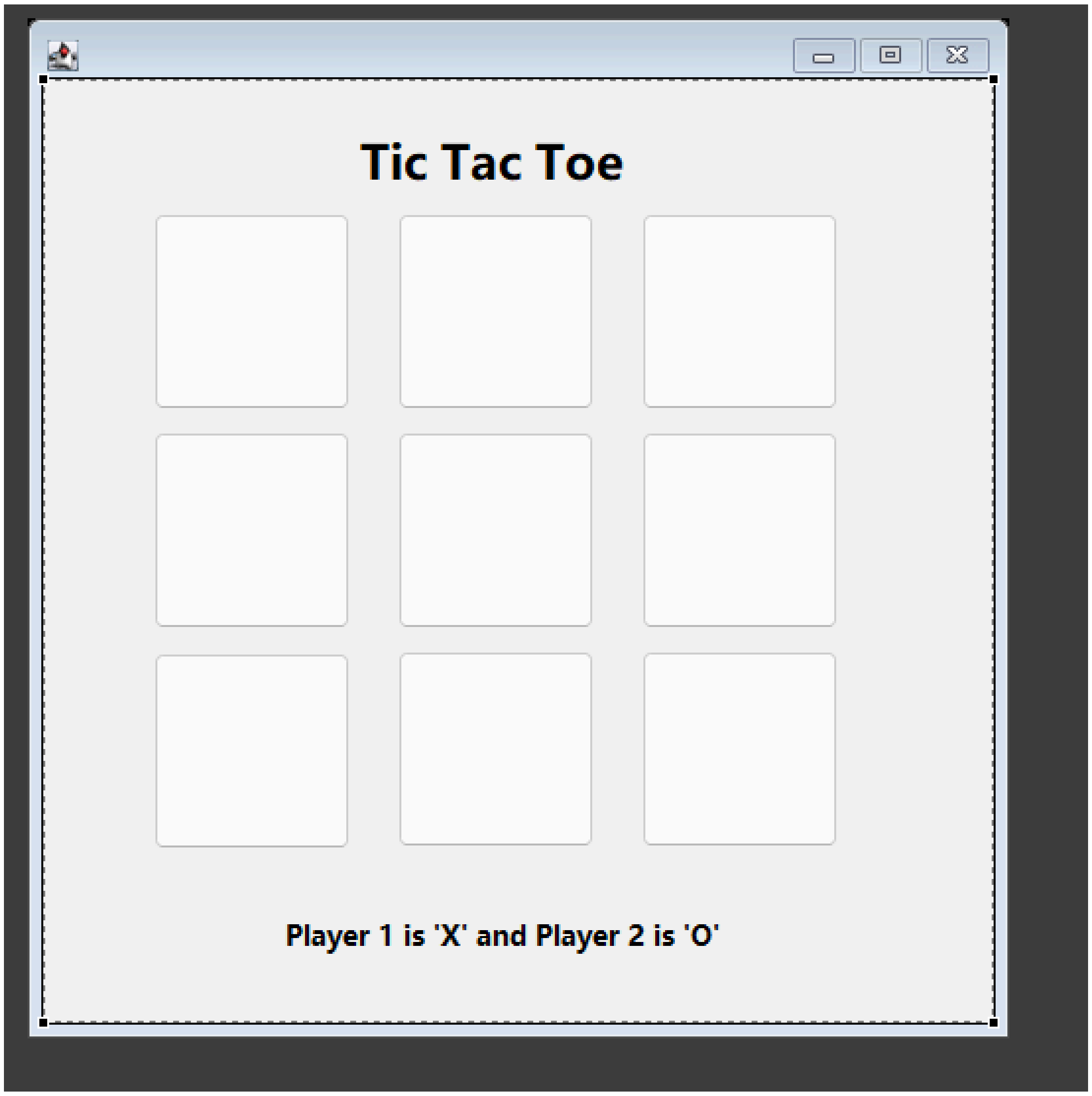
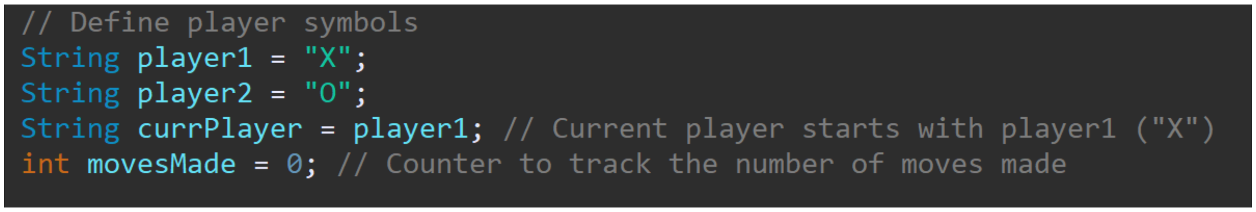
Credit Name: CSE 3010 - Computer Science 3

Assignment: TicTacToeGUI

How has your program changed from planning to coding to now? Please Explain

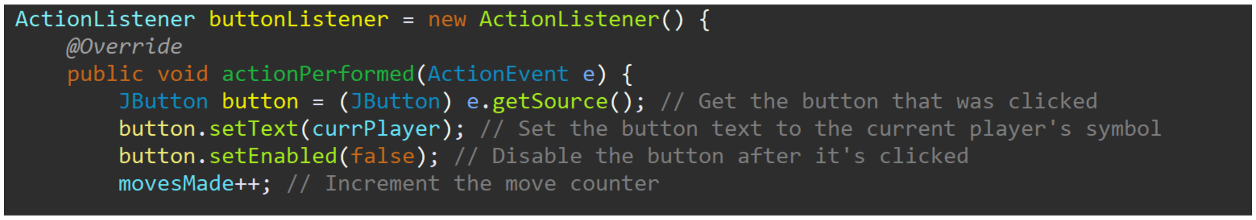


I first designed the board on the design tab. I set up a 3x3 grid made up of JButtons of the same size. Then it is indicated in a JLabel below which player is which. Each JButton was labelled based on its position in the grid. E.g. top left button is labelled btn11 while the top right is called btn13.



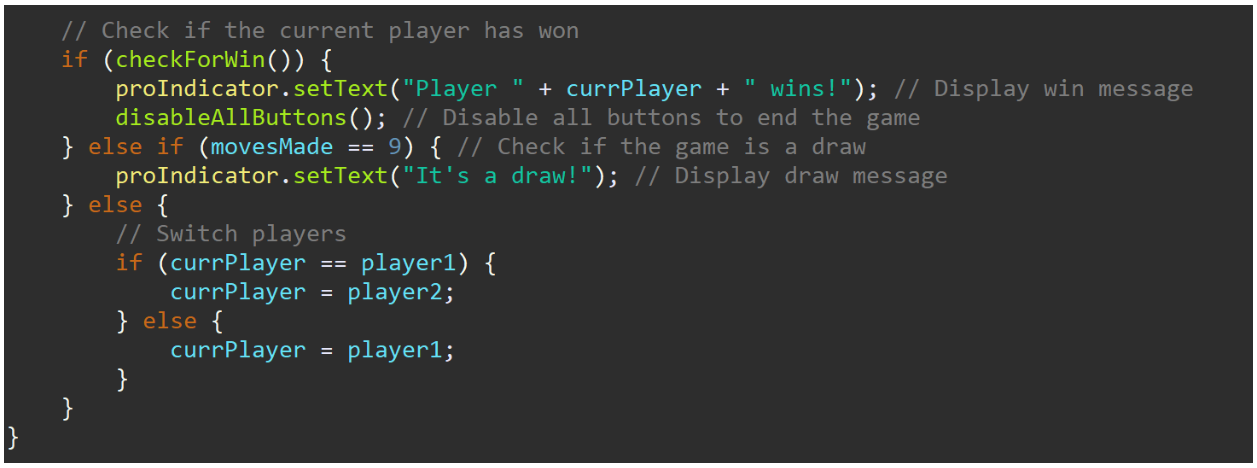
```java
// Define player symbols
String player1 = "X";
String player2 = "O";
String currPlayer = player1; // Current player starts with player1 ("X")
int movesMade = 0; // Counter to track the number of moves made
```

I set up player1 as X and player2 as O as indicated in the GUI. then i made a variable for a current player which is initialized as player1 (X) first by convention and i also made a variable for the number of moves made initialised at 0. These are all instance variables because we will call them in different methods in this code.



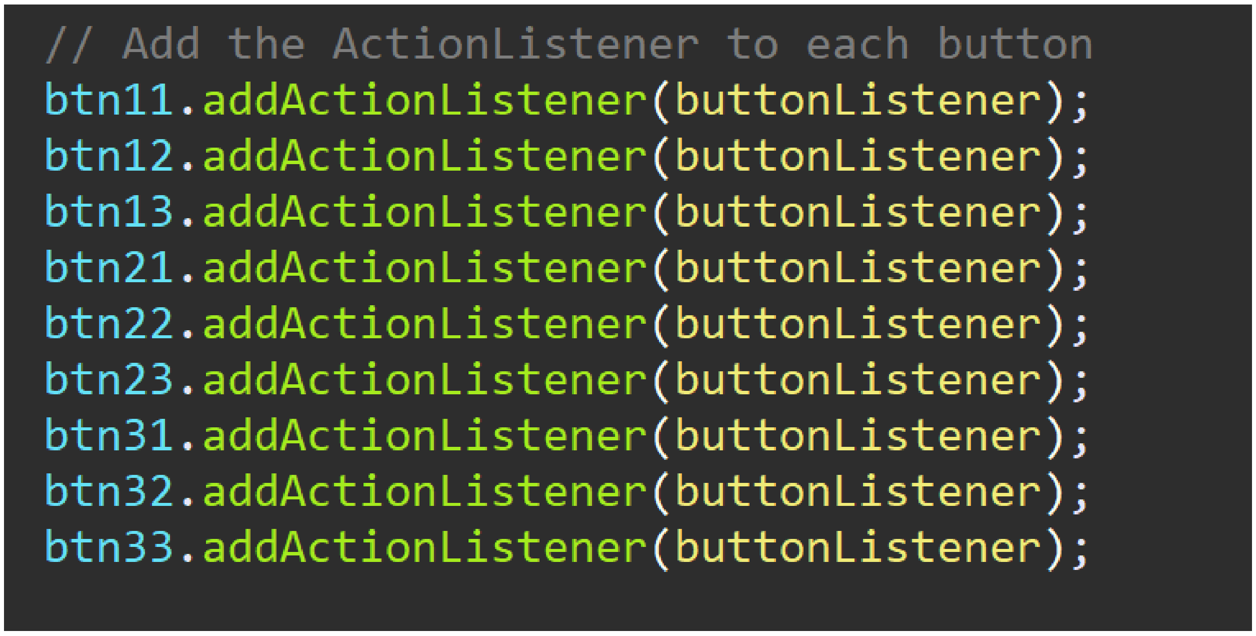
```java
ActionListener buttonListener = new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JButton button = (JButton) e.getSource(); // Get the button that was clicked
        button.setText(currPlayer); // Set the button text to the current player's symbol
        button.setEnabled(false); // Disable the button after it's clicked
        movesMade++; // Increment the move counter
```

I then set up a general button listener which works for all buttons. It works by getting the source from which a button was pressed. And then at this source it sets the text of the button to the current players symbol. Afterwards the button is disabled and the no. of moves made increases by 1.



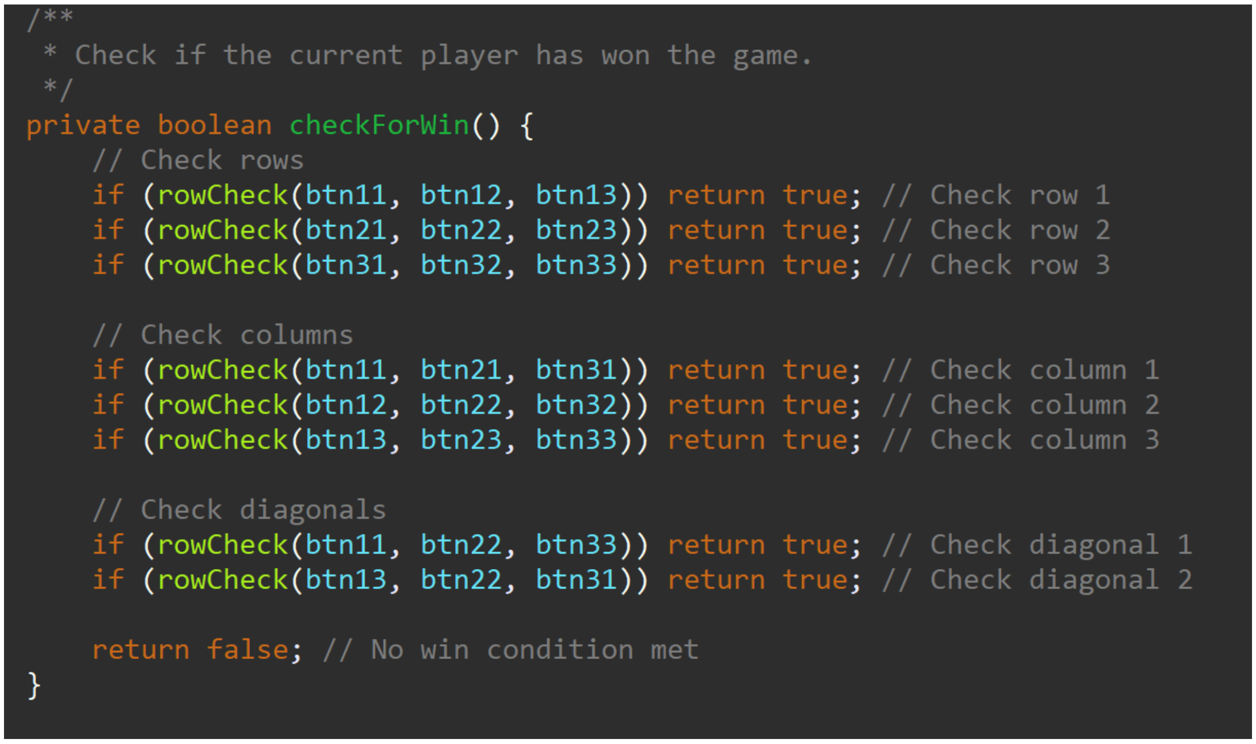
```java
    // Check if the current player has won
    if (checkForWin()) {
        proIndicator.setText("Player " + currPlayer + " wins!"); // Display win message
        disableAllButtons(); // Disable all buttons to end the game
    } else if (movesMade == 9) { // Check if the game is a draw
        proIndicator.setText("It's a draw!"); // Display draw message
    } else {
        // Switch players
        if (currPlayer == player1) {
            currPlayer = player2;
        } else {
            currPlayer = player1;
        }
    }
}
```

If there is a win indicated by the boolean return method checkForWin() then there is a message displayed on the progress indicator(where it was mentioned which player is which). And then all buttons are disabled as the game ended. Otherwise if moves made reaches 9, then its indicated int he progress indicator that the game was a draw. If neither of these options are true then the game is still in progress and we switch the currPlayer as shown in the nested if-else statement.



```java
// Add the ActionListener to each button
btn11.addActionListener(buttonListener);
btn12.addActionListener(buttonListener);
btn13.addActionListener(buttonListener);
btn21.addActionListener(buttonListener);
btn22.addActionListener(buttonListener);
btn23.addActionListener(buttonListener);
btn31.addActionListener(buttonListener);
btn32.addActionListener(buttonListener);
btn33.addActionListener(buttonListener);
```

We add the action listener to every button.



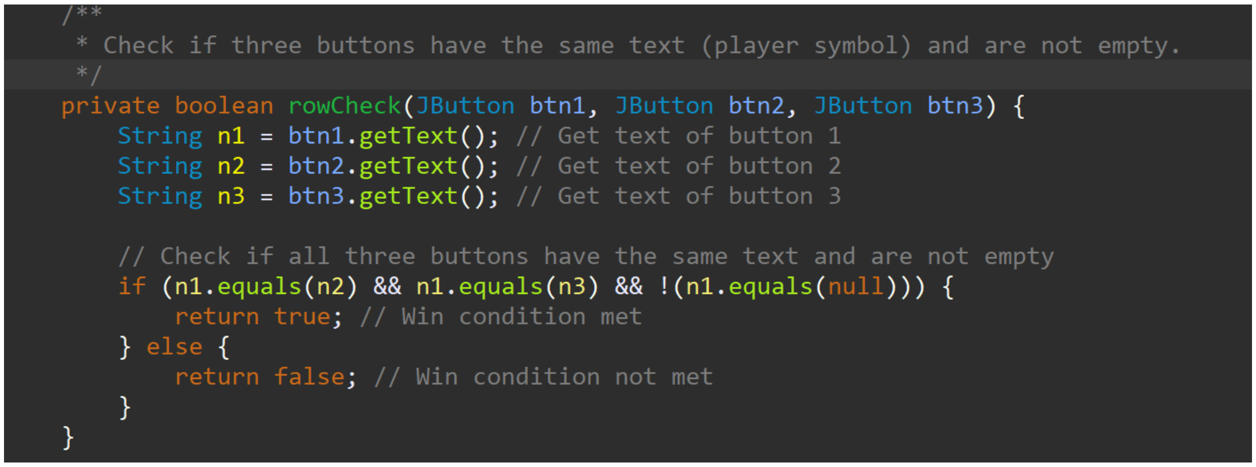
```java
/**
 * Check if the current player has won the game.
 */
private boolean checkForWin() {
    // Check rows
    if (rowCheck(btn11, btn12, btn13)) return true; // Check row 1
    if (rowCheck(btn21, btn22, btn23)) return true; // Check row 2
    if (rowCheck(btn31, btn32, btn33)) return true; // Check row 3

    // Check columns
    if (rowCheck(btn11, btn21, btn31)) return true; // Check column 1
    if (rowCheck(btn12, btn22, btn32)) return true; // Check column 2
    if (rowCheck(btn13, btn23, btn33)) return true; // Check column 3

    // Check diagonals
    if (rowCheck(btn11, btn22, btn33)) return true; // Check diagonal 1
    if (rowCheck(btn13, btn22, btn31)) return true; // Check diagonal 2

    return false; // No win condition met
}
```

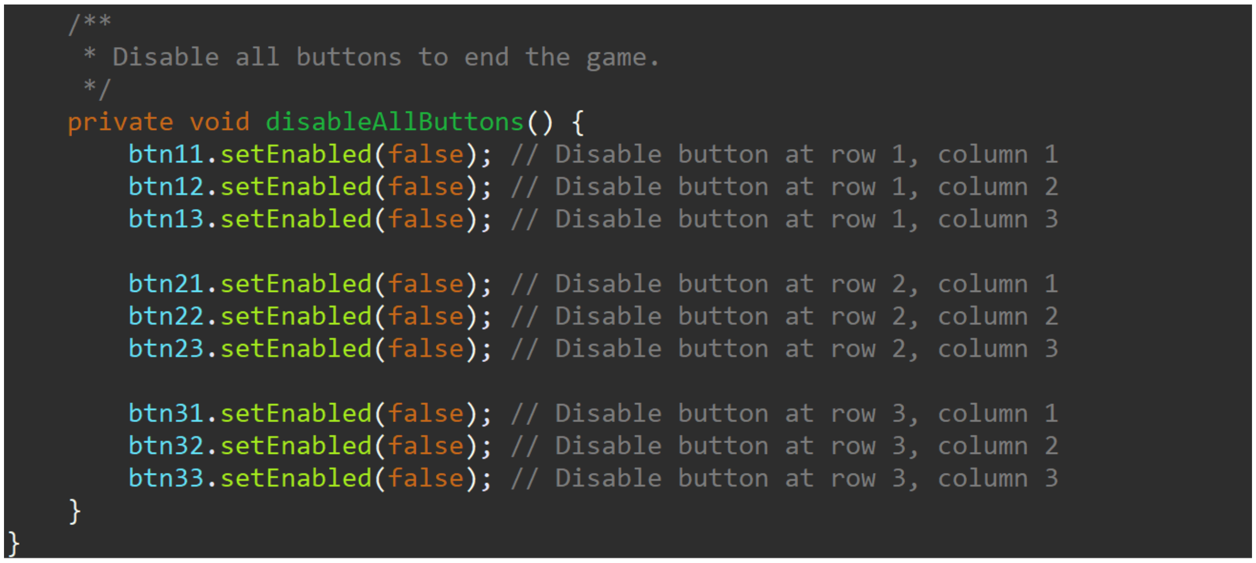
The checkForWin uses another method inside it called rowCheck. But we simply check all the rows, columns and diagonals by using the indexing on the buttons we initially indicated. If no winning row is completed then checkForWin() returns as false.



```
    /**
     * Check if three buttons have the same text (player symbol) and are not empty.
     */
    private boolean rowCheck(JButton btn1, JButton btn2, JButton btn3) {
        String n1 = btn1.getText(); // Get text of button 1
        String n2 = btn2.getText(); // Get text of button 2
        String n3 = btn3.getText(); // Get text of button 3

        // Check if all three buttons have the same text and are not empty
        if (n1.equals(n2) && n1.equals(n3) && !(n1.equals(null))) {
            return true; // Win condition met
        } else {
            return false; // Win condition not met
        }
    }
```

rowCheck is a method which works by taking in the JButtons you want to check, getting their text, and then seeing if they are equal to each other and also not empty. If this is satisfied then true is displayed otherwise its false.



```
    /**
     * Disable all buttons to end the game.
     */
    private void disableAllButtons() {
        btn11.setEnabled(false); // Disable button at row 1, column 1
        btn12.setEnabled(false); // Disable button at row 1, column 2
        btn13.setEnabled(false); // Disable button at row 1, column 3

        btn21.setEnabled(false); // Disable button at row 2, column 1
        btn22.setEnabled(false); // Disable button at row 2, column 2
        btn23.setEnabled(false); // Disable button at row 2, column 3

        btn31.setEnabled(false); // Disable button at row 3, column 1
        btn32.setEnabled(false); // Disable button at row 3, column 2
        btn33.setEnabled(false); // Disable button at row 3, column 3
    }
}
```

the disableAllButtons() simply just disables every button in the grid.