**IT 2045C Computer Programming II   
Prof. Tom Wulf   
Ass 01: GUI Tic Tac Toe Program   
Spring 2015**

Objective: Get practice with using the Java GUI methods.   
 Adding a GUI to an existing console application program  
   
**Discussion:**   
  
 It is not uncommon to develop the logic of a program as a console application (NO GUI) and then graft a GUI on to it later. There are also many reasons for separating the GUI of the program from the rest of the code logic. (This is part of a common design pattern called Model View Controller (MVC) where the GUI represents the View aspect.) For instance, you might have a Web-based or mobile device GUI in addition to a Desktop Application one…

So for this lab, I want you to recode the 2-player Tic Tac Toe game (as we did in CP I) previous lab to use a Java Swing GUI. You can use the same internal data representation (board) and much of the code from the previous version. If you were not with me for CP I, you can get a copy of the Tic Tac Toe console program code from Bb.

The computer does not play the game but facilitates two human players enforcing the rules for the game and determining ties and wins. It should prompt the user to play again when a game ends.

1. Create a java Swing GUI application in a new IntelliJ project called **TicTacToeGUI**. Your project will have a **TicTacToeFrame.java** class and a java main class: **TicTacToeRunner.java**. Put your project under GitHub source control.
2. The game is the same in every respect to the previous lab from CP I (where we refactored Tic Tac Toe console code from the text) except that there is now a GUI to handle the display of the game state and the input from the user. Starting with X each player alternates making a move by clicking on a square. The game blocks until a legal move is entered and then switches to receive input from the other player. The game should check for wins after each move starting with the 5th and should check for a tie (7th move).
3. Use JOptionPane to msg the user as needed for illegal moves and when the game is won or tied, or the user quits. Do not use any console (System.out.print…) output as this is a GUI program.
4. Use grid layout to create a 3 X 3 matrix of JButton objects for the Tic Tac Toe board. Provide a quit button as well. Hint: use an Array of JButton objects for the squares. IF you do this in a clever way as a 2D array, it will interface nicely with your previous code for the game. I suggest that you sub-class the JButton to create a TicTacToeButton class that holds the state of the button. There is a sample example of this TicTacToeTile.java and a test program TicTacToeTileTester.java)
5. You also want to create a single listener instance for all the Buttons on the board. It should determine the row col position of the Button and interface with the code for the game logic.
6. Provide many screen shots that establish that your game works correctly:

* Win dialog:

1. Diagonal win

A screenshot of a computer

Description automatically generated

1. O diagonal wins

A screenshot of a computer

Description automatically generated

* Tie dialogs (Full board tie and the not full board tie)

A screenshot of a computer

Description automatically generated

* Looping until user enters a valid move

I don’t know how to screenshot this

* Prompt to play another game.

A screenshot of a computer

Description automatically generated

1. Submit the link to your GitHub repo and this file renamed **Lastname\_Firstname \_Ass01.docx** (using your actual name and submit your file using the Canvas assignments mechanism.

<https://github.com/DMKALALA/Java-II-/tree/main/Labs/TicTacToeGUI>