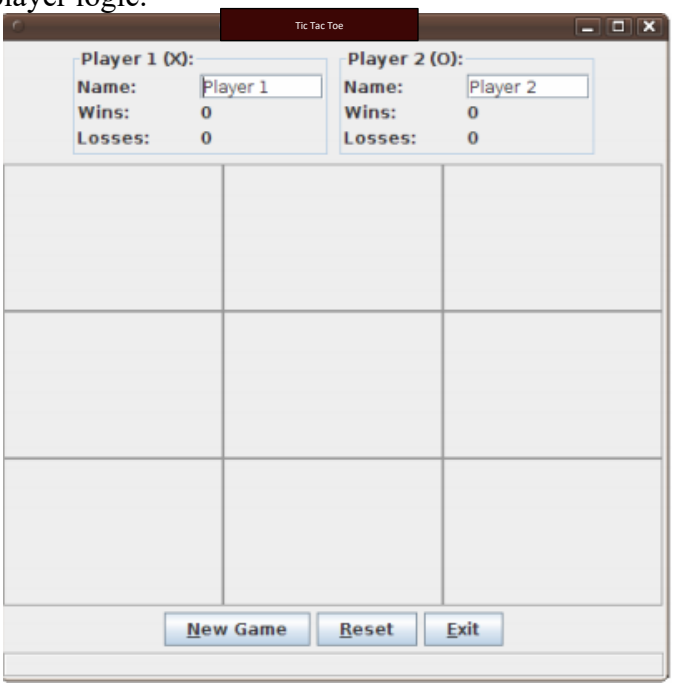
# CPSC 224 Spring 2019
# Homework #3
# Due Date: Friday 22nd 2019 (Midnight)
# TIC TAC TOE

**How to Play:**

Tic-Tac-Toe is a simple game for 2 players played on a 3x3 grid. Each player is given a letter; the first player is "X" and the second player is "O". The players take turns where each places his/her letter in a square of the board until one player wins by placing three of his/her symbol in a line. The line can be horizontal, vertical, or diagonal. The game also ends if the entire board fills with no player completing such a line; which represents a tie.

**Program Description:**

In this assignment you will design and implement a basic graphical user interface for playing a sequence of games of tic-tac-toe. Your program will keep track of game state and information about the two players, including their names and how many games each has won and lost. Both players in the game are controlled by the human users sitting at the computer; you do not need to implement computer player logic.

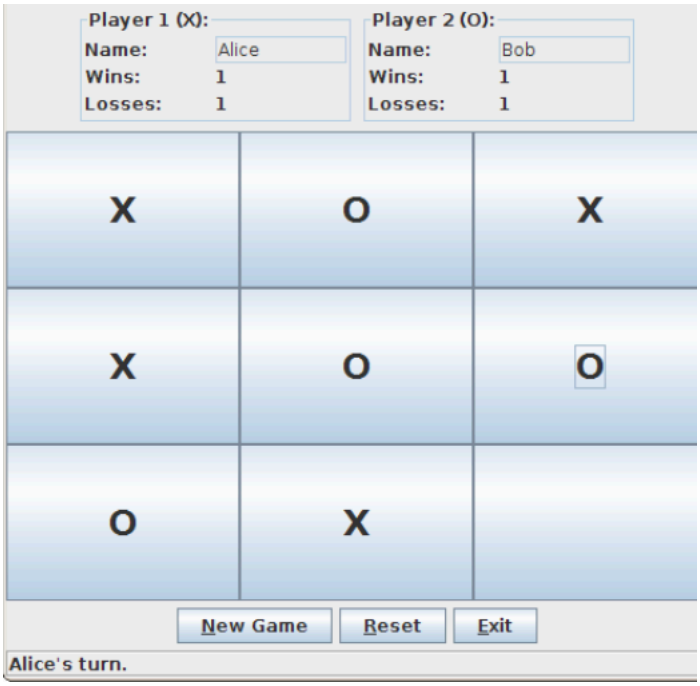



The **GUI window** is of size **500x500 px**. Its top area consists of a pair of top sections about the two game players, including each player's name in an **8-character-wide text field**, wins, and losses. Each player information area appears at its preferred size, just large enough to fit its contents, and is surrounded by a titled border. The text labels for each player at left and right have equal width and are aligned into two columns.

The middle region of the GUI contains a 3x3 grid of nine buttons representing the squares of the game board. The 3x3 grid of buttons is sized to fill all available extra space in the window. The

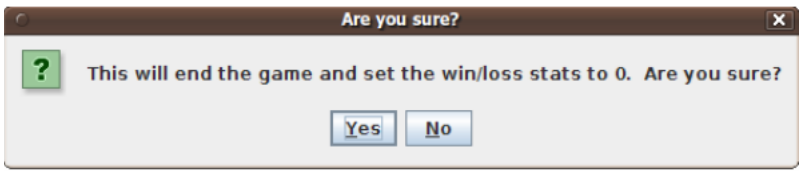
text on each button is shown in a 24-point, bold font, derived from the same font face as the default font for buttons.

Underneath these buttons are three other buttons for starting a New Game round, Resetting the game state, and Exiting the application. These control buttons are centered and sized to their preferred sizes. Under these buttons is a status label stretching horizontally across the bottom of the window that displays various information during and after the game. This label's text is initially "Welcome to Tic-Tac-Toe!". The status label is surrounded by an "etched" style border.




Once the players have typed their names and the user clicks New Game, a game begins. Now the program can be thought of as being in the "game in progress" state. While a game is in progress, the nine game square buttons should enable, allowing the user to click them. A game in progress has a notion of which player's turn it is. The first player, X, always gets to play first. When a player makes a play on an empty square, that player's symbol (X or O) is placed into that square and it becomes the other player's turn. During the game, the status label should always display information about the current player whose turn it is. For example, if the first player's name is Alice and it is her turn, the status label should read, "Alice's turn." While a game is in progress, your GUI should prevent the user from changing the text in the player name fields. If the user clicks on a square that is already occupied, nothing happens.

If the Reset button is clicked, a confirmation dialog box (option pane) appears, asking to confirm the action. It shows the text, "This will end the game and set the win/loss stats to 0. Are you sure?" If the user clicks the Yes button, the program's state should return to exactly the way it was when the program first loads. That is, the player name text fields should become editable and reset their text to "Player 1" and "Player 2"; the win/loss records of both players should reset to 0; any game in progress should stop; the nine game buttons should disable; any game in progress should immediately end; and the bottom status label's text should return to its original state of "Welcome to Tic-Tac-Toe!"




## Implementation

This document does not specify exactly what classes you should write, nor what behavior, fields, methods, etc. each class should have. Part of this assignment is for you to choose appropriate classes and contents for each class based on the specification of desired functionality.

Your program uses a **graphical user interface (GUI)**.You are to implement this GUI using the Java Swing library as taught in class. Your GUI code should be well designed and decomposed with code that is easy to read and modify.

## Hints

It might help you to know that in this program, we interact with the following Swing components: ***JButton, JFrame, JLabel, JOptionPane, JPanel, and JTextField***.

We also interact with other GUI related classes such as *Font, Border, BorderFactory*, various layout managers, and action event listeners. You can enable/disable a component by calling its *setEnabled* method, and you can set whether a text component's text can be edited by calling its *setEditable* method. You can set borders around various components using the **BorderFactory** class.

In part of this program you are expected to pop up a confirmation dialog box to confirm whether the user wants to Reset the game state. Do this using the static methods of class **JOptionPane**. Note that the default confirmation box has three buttons: Yes, No, and Cancel. You should change this to show only Yes and No buttons by passing additional parameters to your method call. In particular, you should pass **JOptionPane.YES_NO_OPTION** as the type of the dialog. Note that the game should reset only if the user clicks "Yes", which corresponds to a value of **JOptionPane.YES_OPTION** being returned by the static method call. Similarly, when the user tries to start a New Game with invalid (empty or allwhitespace) player names, a message dialog should pop up saying that the names were illegal. This message dialog should be shown as an "error message" dialog by passing additional parameters; in particular, pass **JOptionPane.ERROR_MESSAGE** to give the message box a special error icon and appearance.

## Submitting Your Files:

Since we don't know what classes you will choose, you should submit a .zip file named hw3.zip containing all .java source files necessary to build and execute your program. There should be a Makefile as part of your .zip file.
The two lines of linux commands that should be in you Make file are:

```
javac *.java
java TicTacToeMain
```

The files in your ZIP archive should be in the root level directory of the ZIP archive; in other words, when we unzip your file, your .java files should appear in the current directory. Do not include .class, .txt, or other supporting files, only .java.