**Computer Programming I  
Prof. Tom Wulf**

Lab 10: Tic Tac Toe  
**20 pts**

# Objectives:

* Practice working with 2-Dimensional Arrays
* Practice decomposing a complex task into simpler sub-tasks.
* Use an auxiliary file of generalized methods for input
* Use private static methods in the java main class file

You will implement a console based Tic Tac Toe program that allows two users to play the Tic Tac Toe game and then prompts them to play again.

At this point most students are overwhelmed by the challenge of creating the game. However, if you carefully analyze and break down the task into smaller sub-tasks you will be able to do it with a surprising amount of ease. My goal during the sessions this week is to help you get to this point.  
  
Part 1: File of Input Methods

1. Create a IntelliJ project called Lab\_10\_TicTacToe.
2. Now add a copy of your SafeInput.java library to the src folder.   
     
   If you don’t have the SafeInput file and have to recreate it for Tic Tac Toe you will need these methods:

* **public static int getRangedInt(Scanner console, String prompt, int low, int high)**
* **public static boolean getYNConfirm(Scanner console, String prompt)**

1. Notes:

* Make sure the methods are public so they can be used from outside of the file they are in.
* To use the methods within your java main file you reference them by the filename:  
  int rowMove = **SafeInput**.getRangedInt(…);  
  int colMove = **SafeInput**.getRangedInt(…);

# Part 2: java main class

1. Now create a java main class called TicTacToe. This file will have all the main logic for the game.
2. As described in class, you will create a series of helper methods which are only useful for the Tic Tac Toe game. (As opposed to the more general input methods that you can reuse in many programs.) All these helper methods will be **private static** and as we have seen they go within the class (i.e. TicTacToe.java) but not within main().
3. Although it is possible to pass the board[][] array to the helper methods, because they are specific to the program we will declare the board array and any other variables that the helper methods need as **class level** variables so they are directly accessible by the helper methods. (Otherwise we would have to pass the array to each of the functions which is messy.)
4. So, within the class, before main() declare the board array and the constants that define it:  
     
   **private static final int ROW = 3;  
   private static final int COL = 3;  
   private static String board [][] = new String[ROW][COL];**

Be sure to consistently use ROW and COL not the magic number 3 in your code!

1. The helper methods will all go in the main file in the class and after the main() method. You will have to develop your own methods but several that we have already identified in class that you will need are:

* **private static void clearBoard()** // sets all the board elements to a space
* **private static void display() //** shows the Tic Tac Toe game used as part of the promt for the users move choice…
* **private static boolean isValidMove(int row, int col)** // returns true if there is a space at the given proposed move coordinates which means it is a legal move.

**private static boolean isWin(String player)** // checks to see if there is a win state on the current board for the specified player (X or O) This method in turn calls three additional methods that break down the 3 kinds of wins that are possible.

* **private static boolean isColWin(String player)** // checks for a col win for specified player
* **private static boolean isRowWin(String player)** // checks for a row win for the specified player
* **private static boolean isDiagnalWin(String player)** // checks for a diagonal win for the specified player
* **private static boolean isTie()** // checks for a tie condition: all spaces on the board are filled OR there is an X and an O in every win vector (i.e. all possible 8 wins are blocked by having both and X and an O in them.)

1. Create a pseudo code outline for the program using java comments and then code the game.  
     
   Your program should:

* Clear the board and set the player to X (since X always moves first)
* get the coordinates for the move which should be 1 – 3 for the row and col
* convert the player move coordinates to the array indices which are 0 – 2 by subtracting 1
* loop until the converted player coordinates are a valid move
* if appropriate check for a win or a tie (i.e. if it is possible for a win or a tie at this point in the game, check for it.)
* If there is a win or tie announce it and then prompt the players to play again.
* Toggle the player (i.e. X becomes O, O becomes X)

1. Put screen shots at the end of this file to show that you have thoroughly tested the game:

* X wins

Text

Description automatically generated

* O wins

Text

Description automatically generated

* Tie

Text

Description automatically generated

* One showing the re-prompt for a move that is invalid. (i.e. player tries to move to a cell that already has an X or an O in it.

Graphical user interface, text

Description automatically generated

1. Submit this docx file named **Lastname\_Firstname\_Lab10.docx** it should have your screen shots. Use your name.

Then submit either the link to your GitHub repo or your entire IntelliJ project folder in a .zip archive named **Lastname\_Firstname\_Lab10.zip** and of course include this file with your screen shots.Don’t use some other archive format. Follow the directions for naming files, etc.

Resubmit the entire assignment exactly the same way for graduate or extra credit using the link provided.