

**Course:** Principles of Software Development – ENSF 409

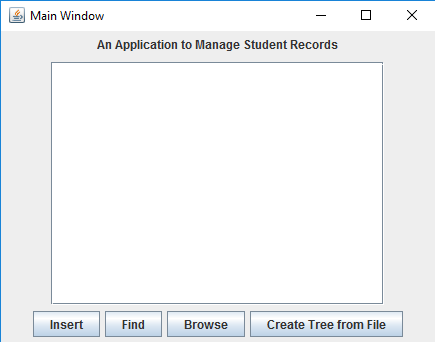
**Lab #:** 7 (Post-lab)

**Student name:** Karan Sunil Bengali

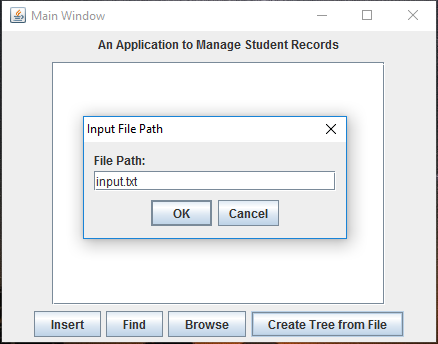
**Date submitted:** March 23, 2017

# Post-Lab Exercise - 1: An Application to Maintain Student Records

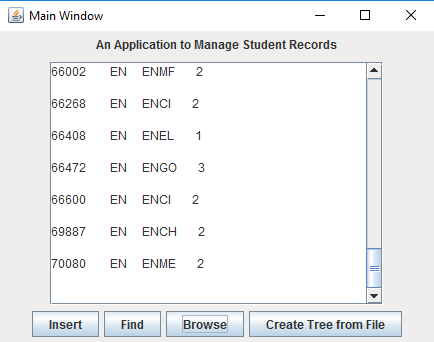
1. Main Window



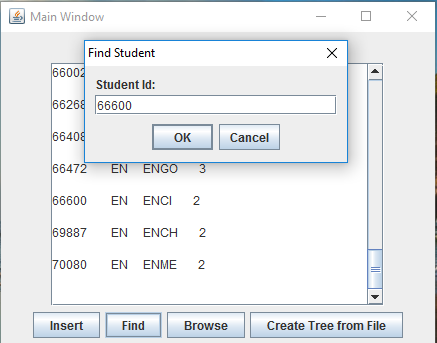
2. When “Create Tree from File” button is pressed a window requesting for file path is displayed



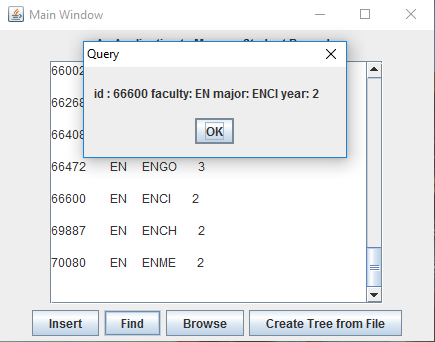
3. Now if we press browse, we can view all the student records.



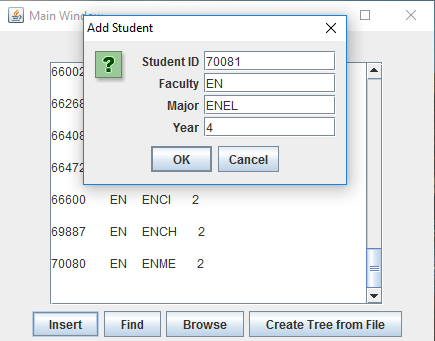
4. Now “find” button is pressed, user can enter the id number of the student they wish to find



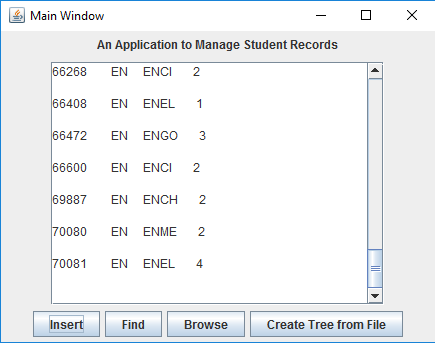
5. When “okay” is pressed, the student will be displayed in a message box if they exist in our search tree.



6. When “insert” is pressed the following frame appears and allows the user to enter new information (student id, faculty, major, year)



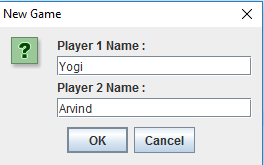
7. When okay is pressed, the student will be added to our binary tree and displayed in the text area.



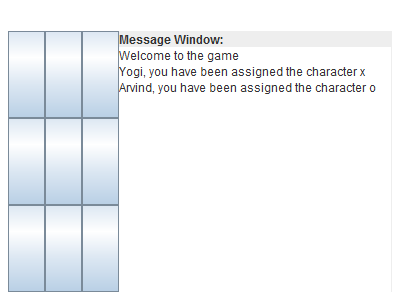
# Post-Lab Exercise - 2: Tic-Tac-Toe with GUI

## Demo:

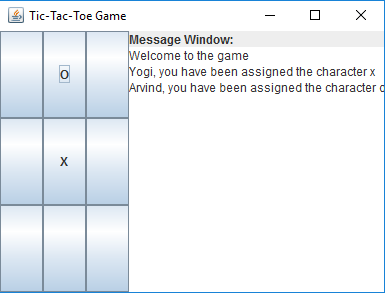
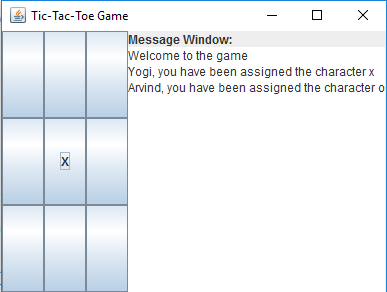
1) Prompt new players to enter their names:

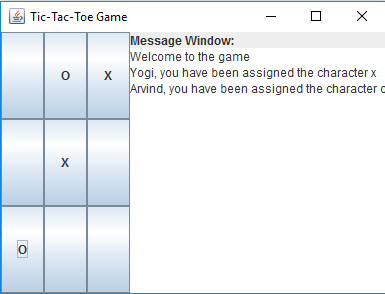
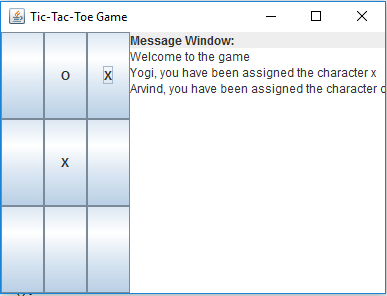


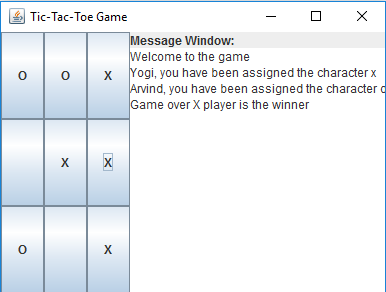
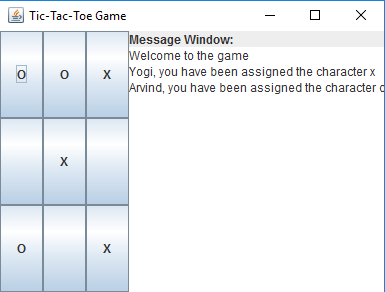
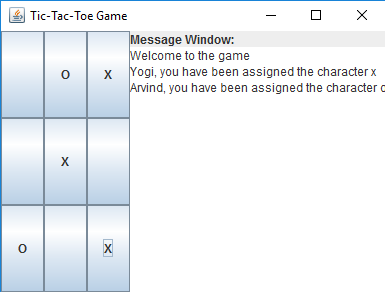
2) Display a welcome message:



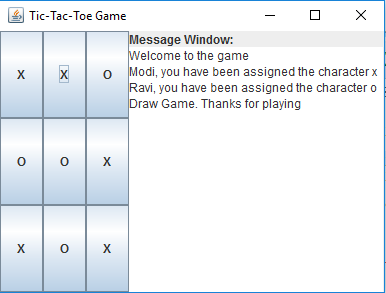
3) Play the game:







4) Games can also result in a draw:



## Source Code:

**package** ex2;

**import** java.io.\*;

**import** javax.swing.JOptionPane;

**import** javax.swing.JTextField;

**public** **class** Game **implements** Constants {

/\*\*

\* The game board

\*/

**private** Board theBoard;

/\*\*

\* The referee

\*/

**private** Referee theRef;

/\*\*

\* Gui that displays the game board

\*/

**private** TicTacToe theGUI;

/\*\*

\* Creates a board for the game

\*/

**public** Game( ) {

theBoard = **new** Board();

theGUI = **new** TicTacToe();

}

/\*\*

\* Calls the referee method runTheGame

\* **@param** r appointed referee for the game

\*/

**public** **void** appointReferee(Referee r){

theRef = r;

theRef.runTheGame();

}

/\*\*

\* Creates new player

\* **@param** board

\* **@param** gui

\* **@return**

\*/

**static** **public** Player create\_players(Board board, TicTacToe gui) {

Player result = **new** Player(board, gui);

**return** result;

}

/\*\*

\* Deafult entry point for the program.

\* Creates players and appoints the referee.

\* **@param** args Command line arguments

\*/

**public** **static** **void** main(String[] args) **throws** IOException {

Referee theRef;

Player player;

Game theGame = **new** Game();

String p1Name = **null**, p2Name = **null**;

JTextField Name1 = **new** JTextField();

JTextField Name2 = **new** JTextField();

Object[] message = {

"Player 1 Name :", Name1,

"Player 2 Name :", Name2,

};

**while**(p1Name == **null** || p2Name == **null**){

**int** option = JOptionPane.*showConfirmDialog*(**null**, message, "New Game" ,JOptionPane.***OK\_CANCEL\_OPTION***);

**if**(option == JOptionPane.***OK\_OPTION***){

p1Name = Name1.getText();

p2Name = Name2.getText();

}

}

theGame.theGUI.textArea.append("Welcome to the game\n");

theGame.theGUI.textArea.append(p1Name + ", you have been assigned the character x\n");

theGame.theGUI.textArea.append(p2Name + ", you have been assigned the character o\n");

player = *create\_players*(theGame.theBoard, theGame.theGUI);

theRef = **new** Referee(player);

theGame.appointReferee(theRef);

}

}

package ex2;

import java.awt.BorderLayout;

import java.awt.GridLayout;

import javax.swing.Box;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JTextArea;

/\*\*

\* Creates the GUI that will display a tic-tac-toe game board.

\*

\* @author Karan Bengali

\* @version 1.0

\* @since March 23, 2017

\*/

public class TicTacToe {

/\*\*

\* Frame that contains panels, buttons, etc.

\*/

JFrame frame;

/\*\*

\* messagePanel - contains the display window for communication

\* boardPanel - contains buttons for the gameboard

\* mainPanel - contains messagePanel and boardPanel

\*/

JPanel mainPanel, messagePanel, boardPanel;

/\*\*

\* messageBox - contains messagePanel

\* boardBox - contains boardPanel

\* mainBox - conatins messageBox and boardBox

\*/

Box mainBox, messageBox, boardBox;

/\*\*

\* The title of our program

\*/

JLabel title;

/\*\*

\* Displays communication messages

\*/

JTextArea textArea;

/\*\*

\* Array of 9 buttons that represent 9 cells on a tic-tac-toe game board

\*/

public JButton [] cellBtns = new JButton[9];

/\*\*

\* Default constructor.

\* 1) Creates messageBox and boardBox

\* 2) Assigns appropriate layout to each

\* 3) Adds panels to each box

\* 4) Assigns a frame size

\*/

public TicTacToe(){

frame = new JFrame("Tic-Tac-Toe Game");

mainBox = Box.createHorizontalBox();

messageBox = Box.createVerticalBox();

boardBox = Box.createVerticalBox();

boardPanel = new JPanel(new GridLayout(3,3));

messagePanel = new JPanel(new BorderLayout ());

messagePanel.add(new JLabel ("Message Window:"), BorderLayout.NORTH);

textArea = new JTextArea(5,15);

messagePanel.add(textArea, BorderLayout.CENTER);

messageBox.add(messagePanel);

frame.setSize(400, 300);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

}

**package** ex2;

/\*\*

\* The purpose of this class is to set up a tick-tac-toe game and start the game.

\*

\* **@author** Karan Bengali

\* **@version** 2

\* **@since** March 23, 2017

\*/

**public** **class** Referee {

/\*\*

\* Public variable player

\*/

**public** Player player;

/\*\*

\* Public variable for the game board

\*/

**public** Board board;

/\*\*

\* Default constructor of the class Referee

\*/

**public** Referee (Player p){

**this**.player = p;

}

/\*\*

\* Function runTheGame will run the game. Player x will start first

\*/

**public** **void** runTheGame(){

player.gamePanel();

}

}

package ex2;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

/\*\*

\* Waits for each player to make a move and determines if it

\* leads to the game finishing in a draw or a win.

\*

\* @author Karan Bengali

\* @version 2

\* @since March 23, 2017

\*/

class Player implements Constants{

/\*\*

\* The game board

\*/

protected Board board;

/\*\*

\* Tic-tac-toe game board GUI

\*/

protected TicTacToe gui;

/\*\*

\* Number of marks on the gameboard

\*/

protected int markCount = 0;

/\*\*

\* Default constructor of class Player. Assigns game board and gui.

\* @param gui tic-tac-toe gui

\* @param boad game board

\*/

public Player (Board b, TicTacToe gui) {

this.board = b;

this.gui = gui;

}

/\*\*

\* Adds 9 buttons to the GUI and assign an action listener to each button

\*/

protected void gamePanel(){

for (int i = 0; i < gui.cellBtns.length; i++){

gui.cellBtns[i] = new MyButton();

gui.boardPanel.add(gui.cellBtns[i]);

}

gui.boardBox.add(gui.boardPanel);

gui.mainBox.add(gui.boardBox);

gui.mainBox.add(gui.messageBox);

gui.frame.add(gui.mainBox);

gui.frame.setVisible(true);

}

/\*\*

\* Implements an actionPerformed method.

\* @author Karan Bengali

\* @version 1.0

\* @since February 23, 2017

\*/

private class MyButton extends JButton implements ActionListener{

/\*\*

\* Default serialized id

\*/

private static final long serialVersionUID = 1L;

/\*\*

\* Variable set to true if a board configuration results in a winner

\*/

boolean win = false;

/\*\*

\* Letter (X or O) to display on the pressed button

\*/

String letter;

/\*\*

\* Default constructor

\*/

public MyButton(){

super();

letter = " ";

setText(letter);

addActionListener(this);

}

/\* (non-Javadoc)

\* @see java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)

\* 1) Wait for a button to be pressed

\* 2) Update the button with the appropriate mark (X or O)

\* 3) Increment the markCounter

\* 4) Check for winner

\* 5) Quit if the game ends in with a winner or in a draw

\*/

@Override

public void actionPerformed(ActionEvent e) {

if((markCount%2) == 0 && getText().charAt(0) == SPACE\_CHAR && !win){

letter = Character.toString(LETTER\_X);

System.out.println(letter + "\n" + markCount);

markCount++;

} else if ((markCount%2) == 1 && getText().charAt(0) == SPACE\_CHAR && !win){

letter = Character.toString(LETTER\_O);

System.out.println(letter + "\n" + markCount);

markCount++;

}

setText(letter);

win = board.checkwinner(gui);

if(win){

gui.textArea.append("Game over " + letter + " player is the winner\n");

} else if (markCount == 9 && !win){

gui.textArea.append("Draw Game. Thanks for playing\n");

}

}

}

}

**package** ex2;

**public** **interface** Constants {

**static** **final** **char** ***SPACE\_CHAR*** = ' ';

**static** **final** **char** ***LETTER\_O*** = 'O';

**static** **final** **char** ***LETTER\_X*** = 'X';

}

**package** ex2;

//STUDENTS SHOULD ADD CLASS COMMENT, METHOD COMMENTS, FIELD COMMENTS

/\*\*

\* This class checks if existing board configuration results in a winner.

\*

\* **@author** Karan Bengali

\* **@version** 2

\* **@since** March 23, 2017

\*/

**public** **class** Board **implements** Constants {

/\*\*

\* Text configurations on buttons that result in:

\* 1) Horizontal wins,

\* 2) Vertical wins, and,

\* 3) Diagonal wins

\*/

**private** **static** **int**[][] *winCombinations* = **new** **int**[][] {

{0, 1, 2}, {3, 4, 5}, {6, 7, 8},

{0, 3, 6}, {1, 4, 7}, {2, 5, 8},

{0, 4, 8}, {2, 4, 6}

};

/\*\*

\* Default constructor

\*/

**public** Board() {

}

/\*\*

\* Checks if the exisiting text configuration on the buttons results in a winner

\* **@param** gui Tic-Tac-Toe GUI

\* **@return** true if configuration results in a winner, false otherwise

\*/

**public** **boolean** checkwinner(TicTacToe gui){

**for**(**int** i = 0; i < 8; i++){

**if**( gui.cellBtns[*winCombinations*[i][0]].getText().equals(gui.cellBtns[*winCombinations*[i][1]].getText()) &&

gui.cellBtns[*winCombinations*[i][1]].getText().equals(gui.cellBtns[*winCombinations*[i][2]].getText()) &&

gui.cellBtns[*winCombinations*[i][0]].getText() != " "){

**return** **true**;

}

}

**return** **false**;

}

}