**CIS 163 Project 2**

**A Super Tic Tac Toe program**

**Due Date**

* At the beginning of the lab; see the schedule, last page of the syllabus

**Before Starting the Project**

* Review Chapters 6 – 7 of the CIS163 book
* Read this entire project description before starting

**Learning Objectives**

After completing this project you should be able to:

* Create a project that uses Model View Controller (MVC) design
* Use a 2-Dimensional array of GUI components
* Use a enum type within a programming project
* Use nested loops to solve complex problems that involve 2-Dim arrays
* Pass and/or return an array of objects to/from methods.

**Project description:** You are to write a Java program that will play TicTacToe for any size board. Normally TicTacToe is on a 3 x 3 board and any 3 in a row for any given column, row, or diagonal wins the game (the normal rules for TicTacToe) . The same is true here, but the board is N x N.

*Steps 1 – 6 can be completed in any order (the ordering is a suggestion). Step 7 – 8 must be completed in sequence. Do not start on Step 7 until steps 1 – 6 are completed.*

**Before you turn in your work: use the** [**Java Style Guide**](http://www.cis.gvsu.edu/studentsupport/javaguide) **to document your project.**

**Step 1: Create an IntelliJ project named “TicTacToePrj”**

* Create a package named: package1 (right click on “TicTacToePrj” and select package)
* Create a class named: SuperTicTacToe (right click on “package1” and select class)
  + Chapter 6 has several examples of creating a main method and associated panel class.
* Create a class named: SuperTicTacToePanel (right click on “package1” and select class)
  + See chapter 6 of your book for a typical example of a panel class.
* Create a class named: SuperTicTacToeGame (right click on “package1” and select class)
  + This class contains all the methods for the game of TicTacToe and is shown in Step **6.**
* Create a class named: GameStatus (right click on “package1” and select enum)
  + See Step 2 for details.
* Create a class named: Cell (right click on “package1” and select enum)
  + See Step 3 for details.

**NO OTHER classes can be created without the instructor’s approval.**

**Step 2: Implement the enum class “GameStatus” using the following:**

There are only 4 possible states the game can be in: {X\_WON, O\_WON, CATS, IN\_PROGRESS}

**Step 3: Implement the enum class “Cell” using the following:**

There are only 3 possible states a Cell can be in: {X, O, EMPTY}

**Step 4: Implement the class named: SuperTicTacToe:**

Using the program found in Chapter 6 of your book as a guide, and create a main method that creates a JPanel object (**SuperTicTacToePanel)**.

**Step 5: Implement the class named: SuperTicTacToePanel:**

* Create the following properties for the SuperTicTacToePanel Panel (more if you wish):

*private JButton[][] board;*

*private Cell[][] iBoard;*

*private JButton quitButton;*

*private ImageIcon xIcon;*

*private ImageIcon oIcon;*

*private ImageIcon emptyIcon;*

*private SuperTicTacToeGame game*;

The JButton variable “board” is a 2-dim array that represents the GUI board the user sees; the variable “iBoard” is a 2-dim array of Cells that is the parameter that is received from the game object and represents the TicTacToe board (more details below); the JButton variable “quitButton” is used to quit the game; and the last three ImageIcon variables represent the icons (X,O,empty) shown on the GUI board.

* In the constructor for SuperTicTacToePanel do the following:
  + Create JPanels as needed so that you have a nice looking GUI display
  + Instantiate the JButton quitButton and add to the Panel
  + Instantiate the ImageIcon for “xIcon”, “oIcon” and “emptyIcon”, these represent X’s, O’s and open space on the board

For example:

*xIcon = new ImageIcon ("x.jpg");*

*// place the file in the project folder.*

* + In one of the JPanels (e.g., somePanel), use GridLayout that is 3 X 3 that represents the board. Also, create listeners for every JButton in the 2-dim array variable board. You will need to use a nested loop, for example:

*for (int row = 0; row < 3; row++)*

*for (int col = 0; col < 3; col++) {*

*board[row][col] = new JButton ("", emptyIcon);*

*board[row][col].addActionListener(listener);*

*somePanel.add(board[row][col]);*

*}*

* + Set the title of the JFrame title to “Super TicTacToe”
  + Create a SuperTicTacToeGameTicTacToe Game object:

*game = new SuperTicTacToeGame();*

* Create a private displayBoard() helper method. In this method, first call the getBoard method within the game class (see step 6) to get the TicTacToe board, and then use a nested loop to set the appropriate icon to the JButtons with in the GUI.
  + For example:

*private void displayBoard() {*

*// NOTE: The following method returns the whole board back to*

*// the panel class. Is this a good idea? Talk with your*

*// instructor about this. Perhaps you could have a method that*

*// returns each cell separately. Is that a better idea?*

*iBoard = game.getBoard ();*

*for (int row = 0; row < 3; row++)*

*for (int col = 0; col < 3; col++) {*

*if (iBoard[row][col] == Cell.O)*

*board[row][col].setIcon(xIcon);*

* Create a private inner class named “ButtonListener” that implements an ActionListener. Then create an actionPerformed method that calls the different methods in the TicTacToeGame class using the game object. For example, call the game.select method when a user clicks a JButton on the board and change the icon to the correct icon. Also, if the user clicks the JButton quitButton, then the program exits after confirmation. The following is some of the code that will be needed within the actionPerformed method:

// Determine which button was selected.

*for (int row = 0; row < 3; row++)*

*for (int col = 0; col < 3; col++)*

*if (jButtonsBoard[row][col] == e.getSource())*

*// tell the game which button was selected.*

*game.select(row,col);*

// Display the board using the private method describe above.

*displayBoard();*

// Determine if there is a winner by asking the game object. (see step 6)

*if (game.getGameStatus() == GameStatus.O\_WON) {*

*JOptionPane.showMessageDialog(null, "O won and X” +*

*“lost!\n The game will reset");*

*}*

**Step 6: Implement the class named: SuperTicTacToeGame:**

This class handles ALL of the game activities, and the following methods and properties must be created

* Create the following properties for the SuperTicTacToeGame class (create more properties if needed):

*private Cell[][] board;*

*private GameStatus status;*

* public SuperTicTacToeGame() A constructor method that initializes the board.
  + For example:

*status = GameStatus.IN\_PROGRESS;*

*board = new Cell[3][3];*

*for (int row = 0; row < 3; r++)*

*for (int col = 0; col < 3; c++)*

*board[row][col] = Cell.EMPTY;*

* public select (int row, int col) this method is called from the SuperTicTacToePanel class and is invoked when the user has selected a JButton with in the 2-Dim array at location row, col. This method either marks a cell as a Cell.X. (i.e., “X”) or Cell.O (i.e., Cell.O).
  + For example:

*board[row][col] = Cell.O;*

* public reset() this method is called from the SuperTicTacToePanel class and it resets the board to a new game.
* public GameStatus getGameStatus() this method is called from the SuperTicTacToePanel class and it determines if a player has won the game after the select method (see above) was called. (The following are standard TicTacToe rules) If there are three in a row (X’s or O’s) for any given column, given row or diagonal then a player has won
  + Return a GameStatus.X\_WON if player “X” has three in a row
  + Return a GameStatus.O\_WON if player “O” has three in a row
  + Return a GameStatus.CATS if all the Cells in the board are not empty and there is no win
  + Return a GameStatus.IN\_PROGRESS if the previous rules do not apply
* A public Cell[][] getboard() this method returns the board to the SuperTicTacToePanel so the panel can display the board to the user.
  + For example:

*return board;*

**Step 7: Add the following functionality to the game**

* Ask the user to enter the size of the board to be used. To accomplish this, you can use JOptionPane.showInputDialog (null, "Enter in the size of the board: “); the board must be greater than 2 and less than 15. Suggestion on implementation: ask the user within the SuperTicTacToe class and pass that information to the SuperTicTacToePanel class via the constructor. If the user enters invalid input (e.g., “abc”, -10) then allow the user to (1) gracefully exit the program without errors using the cancel button OR allow the user to enter in different numbers. Both option should be available to the user.
* After you ask the user the size of the board, ask the user the number of connections to win. If the board size is greater than 3 then the number of connection must be greater than 3 (otherwise the game is not very interesting) If the board size is equal to 3 then the number of connections must be 3. Typically, the number of connections is equal to 3 in normal TicTacToe. Important, the number of connections the user enters must be less than or equal to the size of the board.
* Ask the User ‘who starts the game first? X or O’. To accomplish this, you can use an JOptionPane.showInputDialog (null, "Who starts first? X or O“). You must ask the user within the SuperTicTacToe class and pass that information to the SuperTicTacToePanel class via the constructor. Ask user input only once and start every game the same way (i.e., an X every time or an O every time). Finally, if the user enters invalid input (e.g., “Pizza”, -1) then default to an “X” starting first. Give the user a warning message dialog box that a default of an “X” has been used.

**Step 8: Add the following functionality to the game**

* Add on to the main GUI display one new JButton named Undo. This JButton, when clicked will undo the previous operation. Continued clicking of this JButton will undo the game back to the beginning. Be careful to make sure the correct user turn stays in-sync after each click of the JButton. To implement this functionality you must do the following:
  + Using the class called “Point” found in the API, create an ArrayList<Point> where the ArrayList contains all the previous selected positions (i.e., row, col). In other words, every time a user selects a position, save off that position in the ArrayList.
  + Use the ArrayList for the undo operation.

**Step 9: Make the other player (player O) the computer. See if you can make the computer an excellent tictactoe player. Here are the basic rules that you will need to improve:**

1. Attempt to win the game by attempting to place your “O” somewhere on the board that causes you to win. For example: if the number of connections was 3 to win the game, then attempt to place an “O” in every position to see if your AI can win.
2. Attempt to block the user (“X”) from winning by placing an “O” in every position to see if your AI can block the “X” from winning.
3. If there is not a winning position (i.e., step 1) or a blocking position (i.e., step 2), then attempt to come up with a strategy to place your “O” in such a way to have winning future.

Important: Be sure to handle the cats game and undo buttons in this mode. Also, be very careful here when O (the computer) starts the game. This is tricky.

--------------------------- YOU’RE DONE ☺ -------------------------------

**CIS 163 – Computer Science II**

**Project 2: Tic Tac Toe**

|  |  |
| --- | --- |
| Student Name |  |
| Due Date |  |

|  |  |  |
| --- | --- | --- |
| **Graded Item** | **Points** | **Points Secured and Comments** |
| * Javadoc Comments and Coding Style/Technique * (<http://www.cis.gvsu.edu/studentsupport/javaguide>) * Code Indentation (auto format source code in IDE) * Naming Conventions (see Java style guide) * Proper access modifiers for fields and methods * Use of helper (private) methods * Using good variable names * Header/class comments * Every method uses @param and @return ( 1 sentence after) * Every method uses a /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* separator * Overall layout, readability, No text wrap * Using /\*\* … / for each Instance variable * Has many inner “inner” comments | 10 |  |
| TicTacToePanel class (Step 5) | 25 |  |
| TicTacToeGame class (Step 6) | 25 |  |
| Step 7: Additional functionality   * User enters the board size * Number of connections to win * Who starts first | 5  8  5 |  |
| Step 8: Undo functionality | 10 |  |
| Step 9: Additional functionality   * AI | 12 |  |
| **Total** | **100** |  |

**Additional Comments:**