**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 – 8 of the CIS163 book
* Read this entire project description before starting

**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 (standard rules for TicTacToe). However, the rules have been changed for your program. Diagonals do not matter; only ‘n’ (where n is an integer supplied by the user; see Step 7) in a row for any given column or row is how the game is won. In addition, the board is X by X and the left hand side of the board wraps around to the right hand side. The same is true for the top of the board wraps around to the bottom of the board. It is as if the board has been wrapped around in to a “ball-like” shape. A demonstration of the game will be in done class.

**Learning Objectives**

After completing this project you should be able to:

* 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.

*Steps 1 – 6 should be completed together (the ordering is a suggestion). Step 7 – 10 all add extra functionality. It is suggested that Step 7 be performed first as this will probably be the easiest time to do so, but Step 8 – 10 can be done in any order. Keep in mind that each additional step should not break functionality from any other step!*

**Before you turn in your work: use the Java Style Guide to document your project. (10 pts)**

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

* Create a package named: package1 (right click on “Projects” 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)
* 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 an enum named: CellStatus (right click on “package1” and select enum)
  + See Step 2 for details.
* Create an enum named: GameStatus (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 “CellStatus” using the following code:**

**public** **enum** CellStatus {

***X***, ***O***, ***EMPTY***

}

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

**public** **enum** GameStatus {

***X\_WON***, ***O\_WON***, ***CATS***, ***IN\_PROGRESS***

}

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

Using the programs found 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.**

**Note: This step (i.e., Step 5) only requires the simple version of tic-tac-toe with only a 3 x 3 board. Step 7 allows the user to enter in a different size board as well as specify the number of spaces needed to win.**

* Hint: Remember that SuperTicTacToePanel “is a” JPanel (as mentioned in Step 4)…
* Create the following fields for the SuperTicTacToePanel Panel. You may add more fields if you wish:

*private JButton[][] board;*

*private CellStatus iCell;*

*private JButton quitButton;*

*private SuperTicTacToeGame game*;

*private ImageIcon xIcon;*

*private ImageIcon oIcon;*

*private ImageIcon emptyIcon;*

* + The JButton variable “board” is a 2D array that represents the GUI board the user sees; the variable “iCell” is one CellStatus, and is the parameter that is received from the game object (more details below); the JButton variable “quitButton” is used to quit the game;
* 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 < BDSIZE; row++)*

*for (int col = 0; col < BDSIZE; 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, use a nested loop to set the appropriate icon to the JButtons with in the GUI.
  + For example:

*private void displayBoard() {*

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

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

*iCell = game.getCell(row,col);*

*// TODO: ImageIcon icon = icon for iCell*

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

* 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 ImageIcon. 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 < BDSIZE; row++)*

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

*if (board[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.  
Note: Step 6 continues step 5 and does the simple version of tic-tac-toe with only a 3 x 3 board and requires only 3 spaces in a row or column to win. Step 7 allows the user to enter in a different size board and number of spaces to win.**

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

* Create the following properties for the SuperTicTacToeGame class:

*private CellStatus[][] board;*

*private GameStatus status;*

* + If you wish to add more properties, carefully consider whether they are needed as a property (i.e., minimize number of properties) and whether it is appropriate to have the property here or in another class (e.g., SuperTicTacToePanel). The instructor can help you make these determinations.
* public SuperTicTacToeGame() A constructor method that initializes the board.
  + For example:

*status = GameStatus.IN\_PROGRESS;*

*board = new CellStatus[BDSIZE][BDSIZE];*

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

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

*board[r][c] = CellStatus.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 CellStatus.X. (i.e., “X”) or CellStatus.O (i.e., “O”).
  + For example:

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

* public void 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 indicates if a player has won the game after the select method (see above) was called. If there are three in a row (X’s or O’s) for any given column or given row, 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 CellStatus enums 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 CellStatus getCell(int row, int col) this method returns the CellStatus for a given row and col on the board so the SuperTicTacToePanel can display the correct ImageIcon to the user.
  + For example:

*return board[row][col];*

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

* Ask the user to enter the size of the board to be used as well as the number of connections (row or col) to win. The board size must be greater than 2 and less than 10, and the number of connections to win must be greater than 2 and no larger than the board size. To accomplish this, you can use JOptionPane.showInputDialog method. For example, to ask the user to enter in board size the following could be used:

String x = JOptionPane.showInputDialog (null, "Enter in the size of the board: “);

It is suggested suggest that you ask for this information within the SuperTicTacToePanel class.

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

Make an AI feature. Use the following rules.

1. See if you can win, if so, select that position and win.
2. See if you are going to lose, if so, select the position that blocks that move
3. If rules 1 and 2 are not in play, then develop a tactic to select a position that may help you in the future.

**Step 9: Add the following functionality to the game**

* Add on to the main GUI display one new JButton named: **undoButton**. 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.

**Step 10: Add the following functionality to the game**

* Add on to the main GUI display one new JButton named: **Reset**. This JButton, when clicked will ask the user to enter in a new board size and **who will start first**. The board on the screen must resize itself.

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

**CIS 163 – Computer Science II**

**Project 2: Super Tic Tac Toe**

|  |  |
| --- | --- |
| **Student Name** |  |
| Due Date |  |
| Date Submitted, Days Late, Late Penalty |  |

|  |  |  |
| --- | --- | --- |
| **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 * Javadoc comments for classes, methods, and fields * Regular/inner comments where appropriate | 10 |  |
| SuperTicTacToePanel class | 25 |  |
| SuperTicTacToeGame class | 25 |  |
| Step 7: Additional functionality   * User enters the board size * Enters number of connections to win | 10 |  |
| Step 8: Additional functionality   * AI part | 10 |  |
| Step 9: Additional functionality   * Undo | 10 |  |
| Step 10: Additional functionality   * Reset feature and who starts first | 10 |  |
| **Total** | **100** |  |

**Additional Comments:**