**Exemplary README from a git portfolio**

**TicTacToe**

**Project Summary**

This Program is used to allow two people to play a game of tic tac toe on a normal 3x3 or on a user specified square board. It allows the user to take turns making marks on the board until 1 player has three in a row, col, or diagonal. Once a player has three in a row the game will end and it will announce who the winner is

.

**Running the Program**

To run this program you will need a compiler that can run java programs. Run the files through a complier to use the program

**Classes Documentation**

This program has 5 class 1 of which is gui

**Class: TicTacToe**

This class is used to be the main function initializes the entire program.

**Constructor**

There is no constructor in this class

**Attributes**

| **Attribute** | **Description** |
| --- | --- |
| public static final int DEFAULT\_WIDTH | Labek the default width |
| int width | Used for user inputed width |
| TicTacToeController controller | Initializes the controller |
| JFrame win | Initializes the JFram |

**Methods**

| **Returns** | **Method** |
| --- | --- |
| int | parseInt(Int x) |
|  | *A method used to parse integer values from strings* |
| EventDate | **catch(NumberFormatException e)** |
|  | A method used to catch errors a stop the program if an exception is found |
| String | **setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE)** |
|  | *A method from the the JFrame class used to set the default way to close the program* |
| String | **setVisible(true)** |
|  | A Method used to set some as visible or not. |

**Class: TicTacTieController**

A class used to control how the information effects the view and model and is also how the view and model communicate

**Constructor**

public TicTacToeController(int width)

Creates the controller with values for the width.

**Attributes**

| **Attribute** | **Description** |
| --- | --- |
| private final TicTacToeModel model; | Initializes TicTacToeModel |
| private final TicTacToeView view | Initializes TicTacToeView |
| Int row | Parses data from button to get row in which button is located |
| Int col | Parses data from button to get col in which button is located |
| JButton button | Creates a GUI button |

**Methods**

| **Returns** | **Method** |
| --- | --- |
| void | **public void actionPerformed(ActionEvent e)** |
|  | *Collects the data from user interactions and determines results from the action being made* |
| String | **public String getMarkAsString(int row, int col)** |
|  | *Gets the data from a mark at a specific locations and transfers that data to a string value* |
| String | **toString**() |
|  | *Displays the EventDate as a String.* |

**Class: TicTacToeModel**

A class used to create the model of the board for the game

**Constructors**

public TicTacToeModel() I

Creates a TicTacToeModel with default width

public TicTacToeModel(int width)

Creates a TicTacToeModel with user specified width

**Attributes**

| **Attribute** | **Description** |
| --- | --- |
| public enum Mark | Used to store possible data for spaces on the board |
| public enum Result | Used to store possible results for a finished game. |
| private Mark[][] board; | Creates an Empty gameboard |
| private boolean xTurn; | Used to determine whether it is X’s turn or not |
| private int width | Used to determine the size of the gameboard |
|  |  |

**Methods**

| **Returns** | **Method** |
| --- | --- |
| boolean | **public boolean makeMark(int row, int col)** |
|  | *Used to make a mark at a specified location on the gameboard* |
| boolean | private boolean isValidSquare(int row, int col) |
|  | *Used to determine if the specified square is valid location for a mark* |
| boolean | **private boolean isSquareMarked(int row, int col)** |
|  | *Used to determine if the specified square already has a mark in it* |
| boolean | **public Mark getMark(int row, int col)** |
|  | *Used to determine what a specified square is marked with* |
| result | **public Result getResult()** |
|  | *Used to determine what the current result of the game would be* |
| boolean | **private boolean isMarkWin(Mark mark)** |
|  | *Determines if specified mark wins the game* |
| boolean | **private boolean isTie()** |
|  | *Determines if the specified mark creates a tie* |
| boolean | **public boolean isGameover()** |
|  | *Determines if the game is over* |
| boolean | **public boolean isXTurn()** |
|  | *Determines if it is the x players turn* |
| int | **public int getWidth()** |
|  | *Gets the width of the board* |

**Class: TicTacToeMove**

A class used to store the move selection of the player

**Constructor**

public TicTacToeMove(int row, int col)

determines location of current move from row and col

**Attributes**

| **Attribute** | **Description** |
| --- | --- |
| private final int row | Houses the value of a row |
| private final int col | Houses the value of a col |

**Methods**

| **Returns** | **Method** |
| --- | --- |
| int | **public int getRow()** |
|  | *Returns the value of a row* |
| int | **public int getCol()** |
|  | *Returns the value of a col* |

**Class: TicTacToeView**

This class functions to create GUI elements of the game

**Constructor**

public TicTacToeView(TicTacToeController controller, int width)

creates the view of the Tic tac toe game using the controller and given width

**Attributes**

| **Attribute** | **Description** |
| --- | --- |
| private final TicTacToeController controller | Initializes the Controller |
| private final JButton[][] board; | Creates the buttons that will store the marks |
| private final JPanel squaresPanel; | Creates a jpanel that will store the buttons |
| private final JLabel resultLabel; | Creates a label that will display the result of the game |

**Methods**

| **Returns** | **Method** |
| --- | --- |
| void | **public void updateSquares()** |
|  | Puts mark into current square |
| void | **public void disableSquares()** |
|  | *Disables a square after it is picked* |
| void | **public void showResult(String message)** |
|  | *Shows the result of the game* |
| **void** | **public void clearResult()** |
|  | Clears the result of the game and replaces it with no result |