**Project Overview**

(Anthony and Owyn)

This Tic Tac Toe game is a simple 2 player game that use model-view-controller architecture. Players take turns choosing out of 3x3 grid placing a X or O depending on who the current player is. The game automatically detects a win and draws and has a reset button to play again.

**Class Descriptions**

Player(Interface):

* Methods
  + makeMove(char[][] board): return the “Move”
  + getSymbol(): return the current player symbol
  + getName(): returns player name

AbstractPlayer (Abstract Class) Implements Player:

* Attributes
  + String name
  + Char Symbol
* Methods
  + getSymbol(): return the current player symbol
  + getName(): returns player name
  + MakeMove():left abstract

HumanPlayer Extends AbstractPlayer:

* Attributes
  + Extends AbstractPlayer
* Methods
  + MakeMove(char[][] borad): Returns an empty array (actual move is handled by GameController via GUI input)

GameBorad:

* Attributes
  + char[][] board
  + int movesCount
* Methods
  + isCellEmpty(row, col) – Checks if a cell is unoccupied.
  + placeMove(row, col, symbol) – Places a move if valid.
  + checkWin(symbol) – Checks if the given symbol has won.
  + isDraw() – Checks if the game is a draw.
  + resetBoard() – Clears the board for a new game.
  + getBoard() – Returns a deep copy of the board.

GameLogic:

* Attributes
  + Player CurrentPlayer
  + Player player1, player2
  + GameBoard board
  + boolean playerWon
* Methods
  + makeMove(row, col) – Attempts to place a move and checks validity.
  + checkWin() – Checks if the current player has won.
  + isDraw() – Checks if the game is a draw.
  + switchPlayer() – Alternates turns between players.
  + resetGame() – Resets the board and player turns.

TicTacToeGUI:

* Attributes
  + JButton[][] buttons
  + JLabel statusLabel
  + JButton resetButton
* Methods
  + generateWindow() – Initializes and displays the GUI.
  + updateBoard(board) – Updates the UI to reflect the current board state.
  + showWinner(playerName) – Displays the winner.
  + showDraw() – Announces a draw.
  + clearBoard() – Resets the UI for a new game.

GameController:

* Attributes
  + GameLogic gameLogic
  + TicTacToeGUI view
* Methods
  + onCellClicked(row, col) – Processes a player's move and updates the game state.
  + onResetClicked() – Resets the game when the reset button is pressed.
  + updateView() – Refreshes the UI after each move.

**Implementation Details**

The Tic Tac Toe project is implemented using the following the MVC pattern. The Model consists of GameBoard, which manages the 3x3 grid, and GameLogic, which handles win/draw conditions and turn management. The View (TicTacToeGUI) displays the interactive grid using Swing and updates based on game state changes. The Controller (GameController) bridges the View and Model, processing player clicks and triggering logic updates. Players are modeled polymorphically: the Player interface defines common behavior, AbstractPlayer provides shared attributes, and HumanPlayer implements mouse-driven moves.

**Test Cases**

|  |  |
| --- | --- |
| **Action** | **Expected Output** |
| Player clicks a cell | Cell displays symbol (X/O) correctly |
| Player tries to click a filled cell | Cell does not change; no exception thrown |
| Game ends in a win | Message displays winner |
| Game ends in a draw | Message displays "Draw!" |
| Board resets | Empty board displayed |
| Invalid symbol assignment (dev level) | Throws exception |
| Add new player type (e.g., AI) | System supports with minimal code changes |

**Challenges and Solutions**

One of our initial challenges was with creating the GUI. The Swing library was very unfamiliar at first and so it took about an hour of playing around with it and looking at documentation before we were able to figure things out. We especially struggled with creating the ActionListeners for the buttons.

The other thing we struggled with was getting the proper balance of static and non-static keywords so that our code ran properly. This was caused because the main function needed to be static and so we had to statically reference anything used within main but that then caused some issues because code would get run its dependencies were initialized. But eventually after about an hour of debugging and trying different solutions we were able to find a method that worked by moving main to GameController and initializing GameController in main before generating the game window.