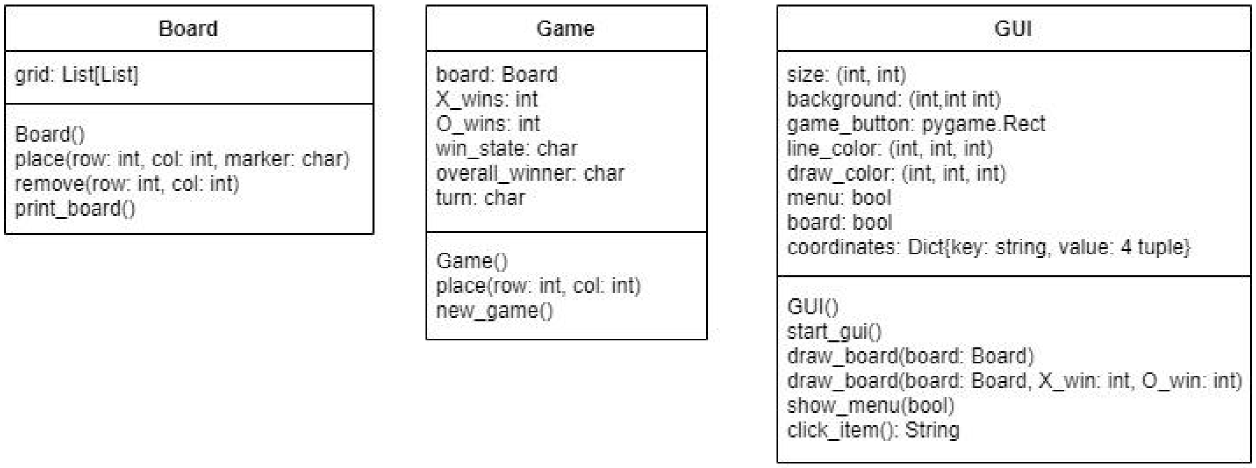

Board():

grid: List[List] - a 3x3 grid representing the tic tac toe board.

place(row: int, col: int, marker: char): Place marker (either X or O) in grid spot marked by row and col. Throws error if marker is not X or O, or if col and row are not in bound.

remove(row: int, col: int): Removes marker at spot specified by row and col.

print_board(): Prints a string representation of the state of the grid.

Game():

board: Board: an instance of the Board class

X_wins: int - Number of x wins.

O_wins: int - Number of O wins.

win_state: char - X, O, D(draw), or N(undetermined). The winner of the current game.

overall_winner: char - X, O or D(draw).

turn: char - X, or O. Turn of the current player

place(row: int, col: int): Place the marker of the player whose turn it is on the grid.

new_game(): Resets the board, win_state, and turn. Everything else is preserved.

Gui():

size: (int, int) - Dimensions of screen

background: (int,int int) - Color of screen

game_button: pygame.Rect - Button of game button

line_color: (int, int, int) - Color of menu lines

draw_color: (int, int, int) - Color of board elements

menu: bool - is menu displayed

options: bool - is options menu displayed?

board: bool - is the tic tac to board displayed?

coordinates: Dict{key: string, value: 4 tuple} - listing of screen elements and locations

start_gui(): Initialize and start the screen

draw_board(board: Board): Displays the provided board parameter to the screen.

draw_board(board: Board, X_win: int, O_win: int): Displays the provided board parameter to the screen, and the number of x and y wins

show_menu(bool): True if the game menu is displayed

click_item(): String: Returns the name of the element the mouse is over.

List of return strings:

Game Button:	“game_button”
New game:	“new”
1 of 1:	“1of1”
2 of 3:	“2of3”
3 of 5:	“3of5”
4 of 7:	“4of7”
Player vs. Player:	“pvp”
Player vs. PC:	“pvpc”
Easy:	“easy”
Hard:	“hard”
Quit!:	“quit”
Grid spot 0, 0	“0,0”
Grid spot 0, 1	“0,1”

● Grid spot 0, 2	“0,2”
● Grid spot 1, 0	“1,0”
● Grid spot 1, 1	“1,1”
● Grid spot 1, 2	“1,2”
● Grid spot 2, 0	“2,0”
● Grid spot 2, 1	“2,1”
● Grid spot 2, 2	“2,2”

Main Program

Needs:

Store

- game settings
    a. Opponent option
        i. Difficulty if PC
    b. Number of games
    c. (Optional: color layout)
- Game instance

Handle events

- Mouse click
    - Respond to board component clicked
- Possibly mouse position (if we implement option highlighting).

Update

- Game component
- GUI component (the GUI only displays what it is told to ,and returns the name of element, it does not automatically do anything)

Compute run the computer player