**Tic Tac Toe:**

**Game-Board Class:**

**Purpose:**

Represents the game board for a Tic-Tac-Toe game.

Properties:

* board[3][3]: A 3x3 array of characters (char). Each element represents a cell on the game board, initially set to Blank.

Behaviors:

* Constructor (GameBoard()): Initializes all cells on the board to Blank.
* draw(): Prints the current state of the game board to the console. It iterates through each cell in the board array and prints its content followed by a space. After printing each row, it outputs a newline character to start a new line for the next row.
* isTaken(int row, int col): Checks if a cell at the specified row and column is already marked with a player's symbol (X or O). Returns true if the cell is taken, otherwise false.
* setX(int row, int col): Attempts to place the X symbol in the specified cell if it's not already taken. Returns true if successful, otherwise false.
* setO(int row, int col): Similar to setX, but attempts to place the O symbol. Also returns true on success and false otherwise.
* isLine(): Checks if there's a winning line (three consecutive Xs or Os) horizontally, vertically, or diagonally. Returns true if there's a win, otherwise false.

Player Class:

Purpose: Represents a player in the Tic-Tac-Toe game.

Properties:

* symbol: Stores the player's symbol, either X or O.

Behaviors:

* Constructor (Player(char symbolChar)): Takes a character as an argument and initializes the symbol property accordingly. Uses static\_cast to convert the character to the Symbol enum type.
* getSymbol(): Returns the player's symbol.
* switchSymbol(): Changes the player's symbol from X to O or vice versa.

Game Class

Purpose: Orchestrates the gameplay, managing turns between players and determining the game's outcome.

Properties:

* board: An instance of GameBoard representing the game's board.
* player1 and player2: Instances of Player, representing the two players.

Behaviors:

* Constructor (Game(Player player1, Player player2)): Initializes the game with two players.
* play(): Executes the game loop. Continues until there's a winning line or the board is full. For each turn:
  + Draws the current board state.
  + Prompts the current player for their move.
  + Validates the move and updates the board if valid.
  + Checks if there's a winning line after each move.
  + Switches the current player after each move.

Main Function

Purpose: Sets up the game environment, creates instances of Player and Game, and starts the game.

Behavior:

* Creates two Player instances, one for X and another for O.
* Creates a Game instance with the two players.
* Calls the play() method on the Game instance to start the game.

Additional Notes

* The game does not currently check for a tie (when the entire board is filled without a winner).
* The game assumes valid input from the user. Improvements could include input validation to ensure users enter valid row and column numbers.
* The Player class's Symbol enum is defined within the class, which is a common practice for encapsulation. However, it's accessible because it's public.

This design provides a solid foundation for a Tic-Tac-Toe game, separating concerns into distinct classes for modularity and ease of maintenance.

Notes on Class Inheritance

In this design, there is no class inheritance. Each class represents a distinct entity in the game's architecture: the game board, individual players, and the game itself. Class inheritance could potentially be introduced if we were to extend the game to include more complex features like different types of boards or special rules for certain players, but for the current scope of a basic Tic-Tac-Toe game, inheritance is not necessary.

Class Diagram:

|  |
| --- |
| GameBoard:  Constructor: GameBoard  Draw  isTaken  setX  setO  isLine |

|  |
| --- |
| Player:  Player  getSymbol  switchSymbol |

|  |
| --- |
| Game:  Game  Play |