Design a Tic-Tac-Toe Game

- Problem Statement
- Overview
- Requirements
- Components
  - 1. PieceType
  - 2. PlayingPiece
  - 3. Board
  - 4. Player
  - 5. TicTacToeGame
- Class Diagram
- Implementation
  - Output

Resources

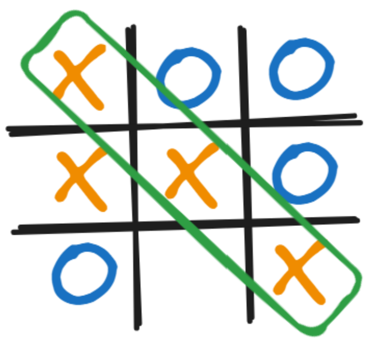
- 7. Design Tic Tac Toe game (Hindi) | Tic-Tac-Toe LLD Java | Low Level Design, System Design
- Code Repo → src/main/java/com/conceptcoding/interviewquestions/tictactoe · main · shrayansh jain / LLD-LowLevelDesign · GitLab

Problem Statement

Design a comprehensive 2-player tic-tac-toe game of Xs and Os with the executable code following all design principles, patterns, best practices and guidelines discussed before.

Overview

Tic-tac-toe is a simple two-player paper-and-pencil game. Each player selects a piece before the game and takes turns placing it on the 3x3 grid. A player wins by placing three pieces in a row, column, or diagonal. The game ends when a player wins or when all grid cells are filled, resulting in a draw.



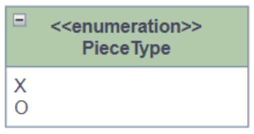
TicTacToe Winner: PlayerX

Requirements

- A 3x3 board should be used to play the game.
- 2 Players are marked by the piece they choose to play with - PlayerX and PlayerO
- The game should end when a player wins.
- The game should end when it's a draw(the players are out of free cells to play).
- The game should not allow any invalid moves.

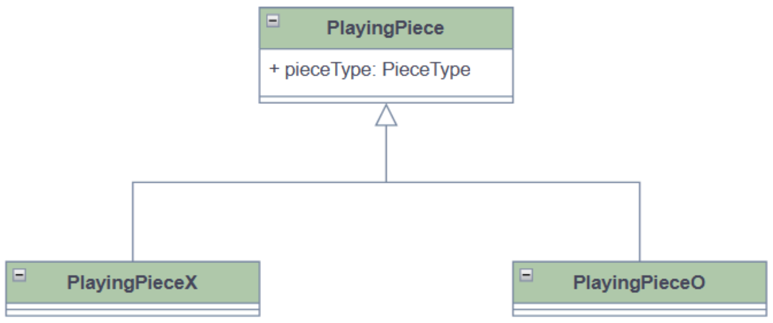
Components

1. `PieceType`



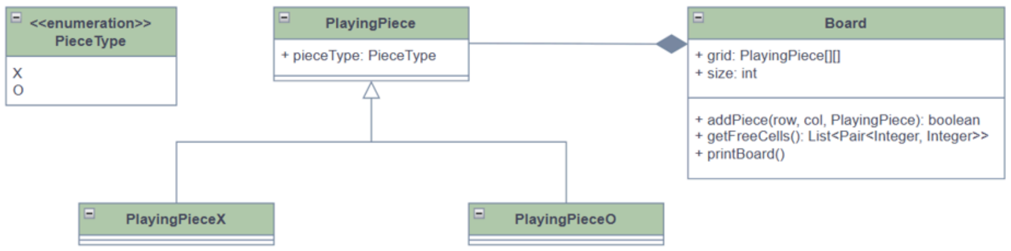
- It's a symbol that is used to represent the cell value.
- Here we are using X and O. We can also choose different symbols.

2. `PlayingPiece`



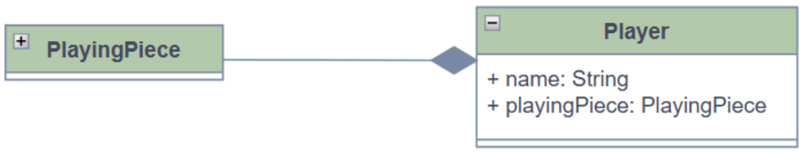
- The base class represents `PlayingPiece,` used to represent the symbol used.
- Holds a reference to `PieceType` used to represent a piece.
- The concrete classes PlayingPieceX and PlayingPieceO denote specific `PlayingPiece`s associated with the corresponding `PieceType`.

3. `Board`



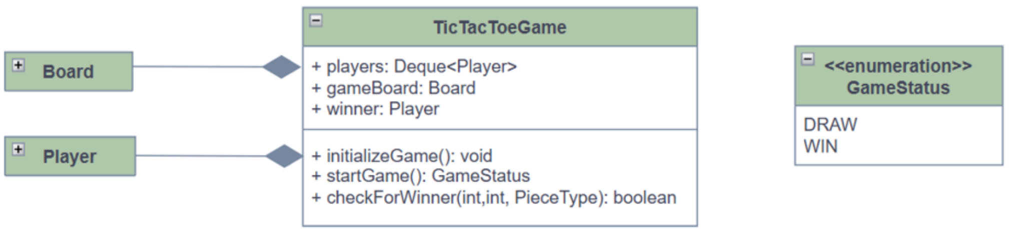
- `Board` is a concrete class that contains a 3x3 matrix of `PlayingPiece` used for playing the game and marking the positions where players place their pieces.
- We can create a grid of any size we want and define corresponding rules to extend the game in future.
- Includes methods to play the game, such as placing a piece, updating the cell value, checking for free cells, and detecting a winning move or game end.

4. `Player`



- `Player` is a concrete class that represents a Player in the game.
- It is composed of a `PlayingPiece` type, a symbol that a `Player` chooses to represent at the beginning of the game.

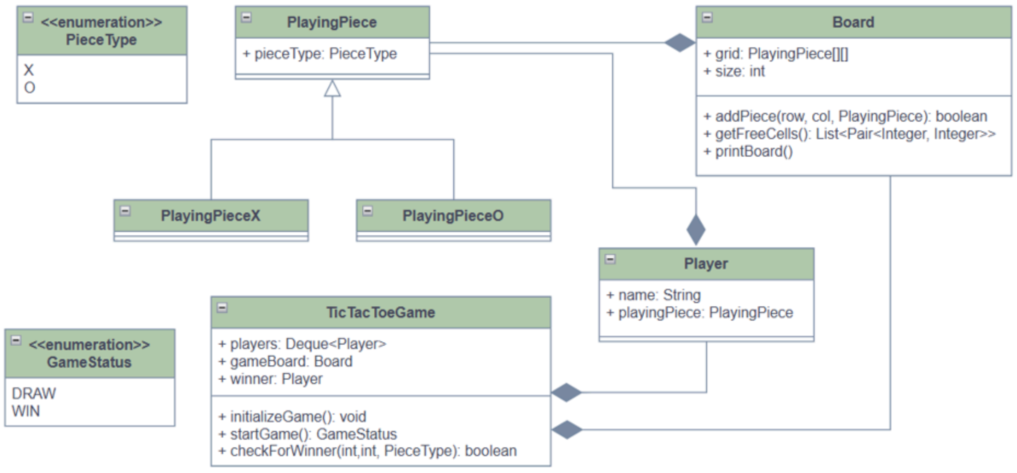
5. `TicTacToeGame`



- This concrete class contains core Game control logic.
- This class handles alternating turns between players, validating their moves, checking for a winning move, displaying the board after each step, and declaring the game winner or a draw before ending the game.
- Hence, it is composed of `Players` and a game `Board` reference for orchestrating the `Game` flow.
- Once the Game is over, the appropriate `GameStatus` value is returned.

Class Diagram

We combine the components above to produce a final solution to the TicTacToe Game Design problem.

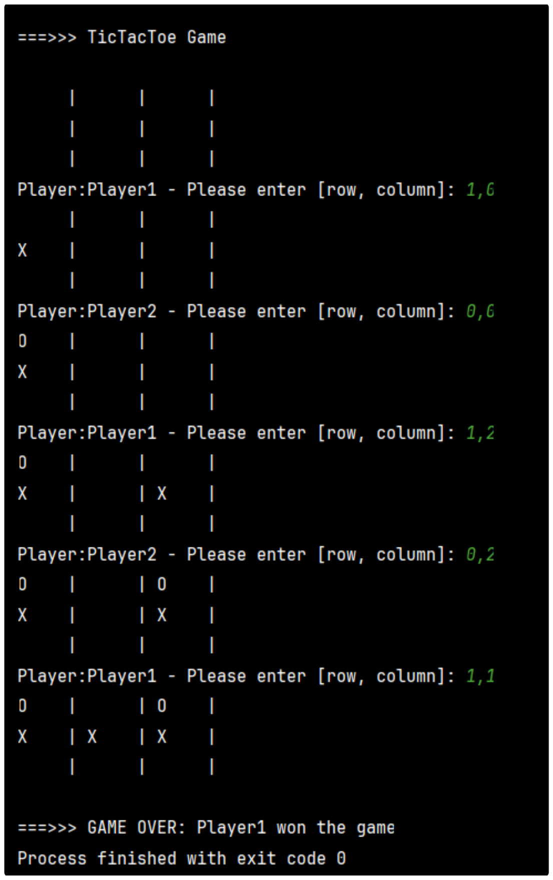


<<enumeration>> PieceType
X
O
PlayingPiece
+ pieceType: PieceType
Board
+ grid: PlayingPiece[][]
+ size: int
+ addPiece(row, col, PlayingPiece): boolean
+ getFreeCells(): List<Pair<Integer, Integer>>
+ printBoard()
PlayingPieceX
PlayingPieceO
Player
+ name: String
+ playingPiece: PlayingPiece
<<enumeration>> GameStatus
DRAW
WIN
TicTacToeGame
+ players: Deque<Player>
+ gameBoard: Board
+ winner: Player
+ initializeGame(): void
+ startGame(): GameStatus
+ checkForWinner(int,int, PieceType): boolean

Implementation

Refer Code Repository for Executable Code → src/main/java/com/conceptcoding/interviewquestions/tictactoe · main · shrayansh jain / LLD-LowLevelDesign · GitLab

Output



```
===>>> TicTacToe Game

     |      |      |
     |      |      |
     |      |      |
Player:Player1 - Please enter [row, column]: 1,0
     |      |      |
X    |      |      |
     |      |      |
Player:Player2 - Please enter [row, column]: 0,0
O    |      |      |
X    |      |      |
     |      |      |
Player:Player1 - Please enter [row, column]: 1,2
O    |      |      |
X    |      | X    |
     |      |      |
Player:Player2 - Please enter [row, column]: 0,2
O    |      | O    |
X    |      | X    |
     |      |      |
Player:Player1 - Please enter [row, column]: 1,1
O    |      | O    |
X    | X    | X    |
     |      |      |

===>>> GAME OVER: Player1 won the game
Process finished with exit code 0
```