Design Documentation

# Class Structure

## GeneralBoard

An interface that contains abstract methods for displaying the board, moving pieces, and determining whether the piece's position is valid.

## TicTacToeBoard

Implements the GeneralBoard interface. Initialize the board layout, display the board, record the piece placement, and check for win or draw conditions.

## SuperBoard

Inherits from GeneralBoard, the initialization and display methods have been overridden.

## QuoridorBoard

Implements the GeneralBoard interface. In addition to implementing the abstract methods in the interface, it also initializes the board, checks whether walls can be placed, and uses BFS to determine if a player is completely trapped.

## GeneralGame

Abstract general game class. Set up methods to handle draws, methods to print the summary table, and methods to reset team statistics.

## TicTacToeGame

Inherits from the GeneralGame class. Show Tic-Tac-Toe game logic, including features such as replaying the game, inputting or changing the board size, entering the current player's number, allowing players to choose their piece type, updating and displaying the board, determining and recording the outcome, and switching players.

## OrderAndChaosGame

Inherits from the TicTacToeGame class. Show Order and Chaos game logic, which is similar to Tic-Tac-Toe, but the `inputPlayer` and `choosePiece` methods have been overridden. This is because the team names are fixed as 'ORDER' and 'CHAOS', and players can choose either 'X' or 'O' in each round.

## SuperTicTacToeGame

Inherits from the TicTacToeGame class. Show Super Tic-Tac-Toe game logic. I have divided the super board into 9 mini boards, first checking the victory conditions for the mini boards before placing them into the super board for further checks.

## QuoridorGame

Inherits from the TicTacToeGame. Show quoridor game logic, including choosing player numbers(2 or 4), initializing players, setting walls , making movements, checking wining situations, switching players, printing summary table and continuing the game. The reason why extending the TicTacToeGame because they have the similar game flow.

## Player

Record the relevant members required for the player.

## Team

Record the relevant members required for the team.

## Piece

Record the piece type.

## Tile

Record the piece status on the board cell.

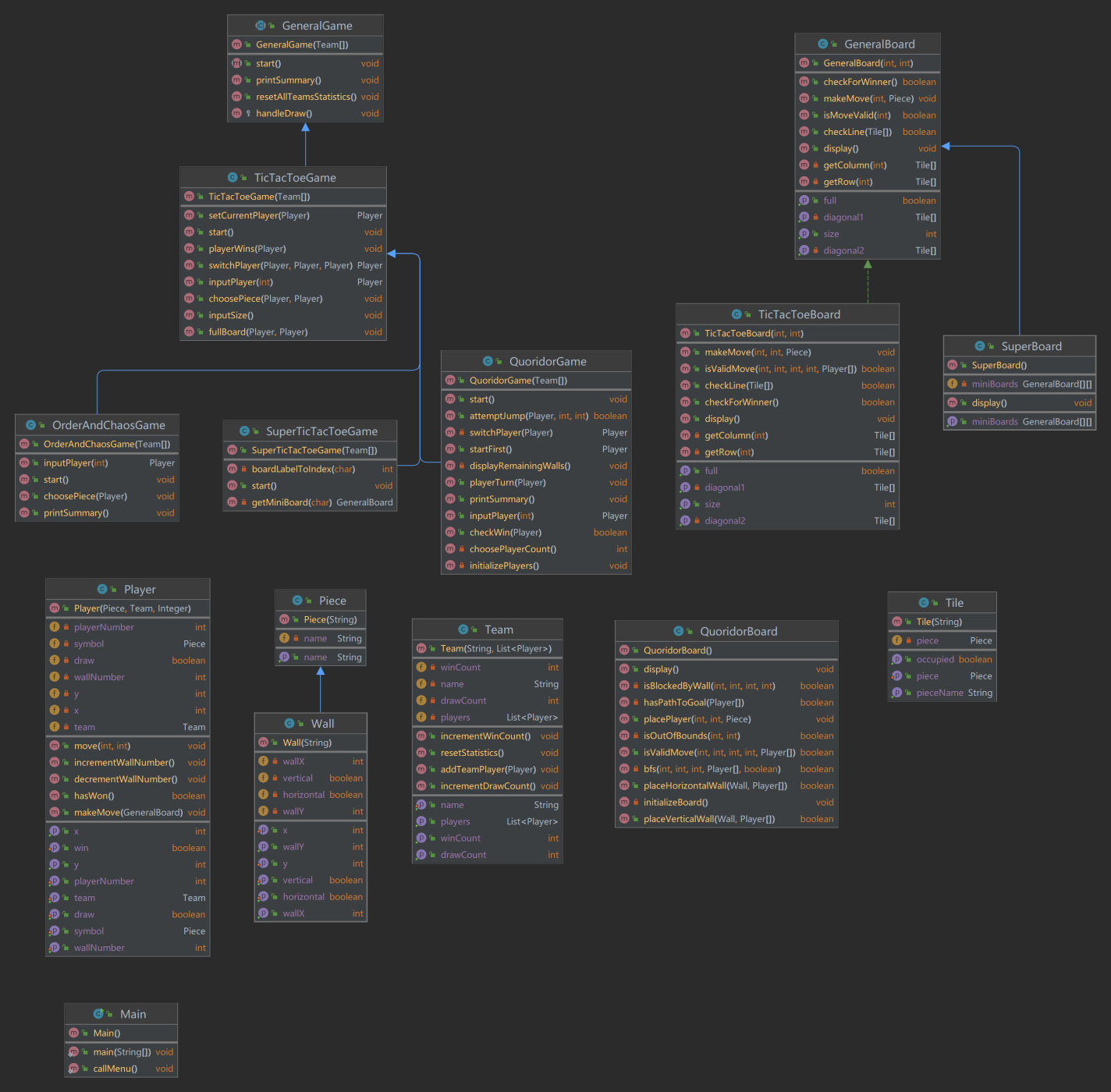
## Wall

Inherits from Piece where the status of the walls are stored.

## Main

Initialize and start the game.

# UML graph



# Evaluation

* This design document describes a scalable and extensible turn-based game framework. The framework supports various turn-based game variants such as tic-tac-toe, super tic-tac-toe, Chaos and Order, and Quoridor. By employing object-oriented design principles, we have created a flexible system that allows for the easy addition of new game variants without major changes to the existing code.
* Scalability and Extendibility: By utilizing the abstract class GeneralGame and the general class GeneralBoard, we have created a flexible foundation that can be easily extended to support new game variants. Each specific game class inherits from GeneralGame and implements its unique game logic. This design allows us to add new games without modifying the existing code.The QuoridorBoard class extends the GeneralBoard and adds logic specific to the Quoridor game. By extending it, we can reuse the general board logic from GeneralBoard while incorporating features unique to Quoridor.
* Our design allows for easy expansion of existing games. The QuoridorGame class inherits from TicTacToeGame and adds game logic specific to Quoridor. By using inheritance, we can reuse the general game logic from TicTacToeGame while incorporating features unique to Quoridor.
* In order to allow for another "turn based" variant to be playeed. The current Tile class uses String types to represent positions and piece names, which can lead to type safety issues. It may be worth considering using enumerations or other types to enhance type safety. We also need to create new game classes and a game board class, add an option for new games in the menu, and develop new types of game pieces. Finally, we should abstract common game logic so that different games can share this logic.