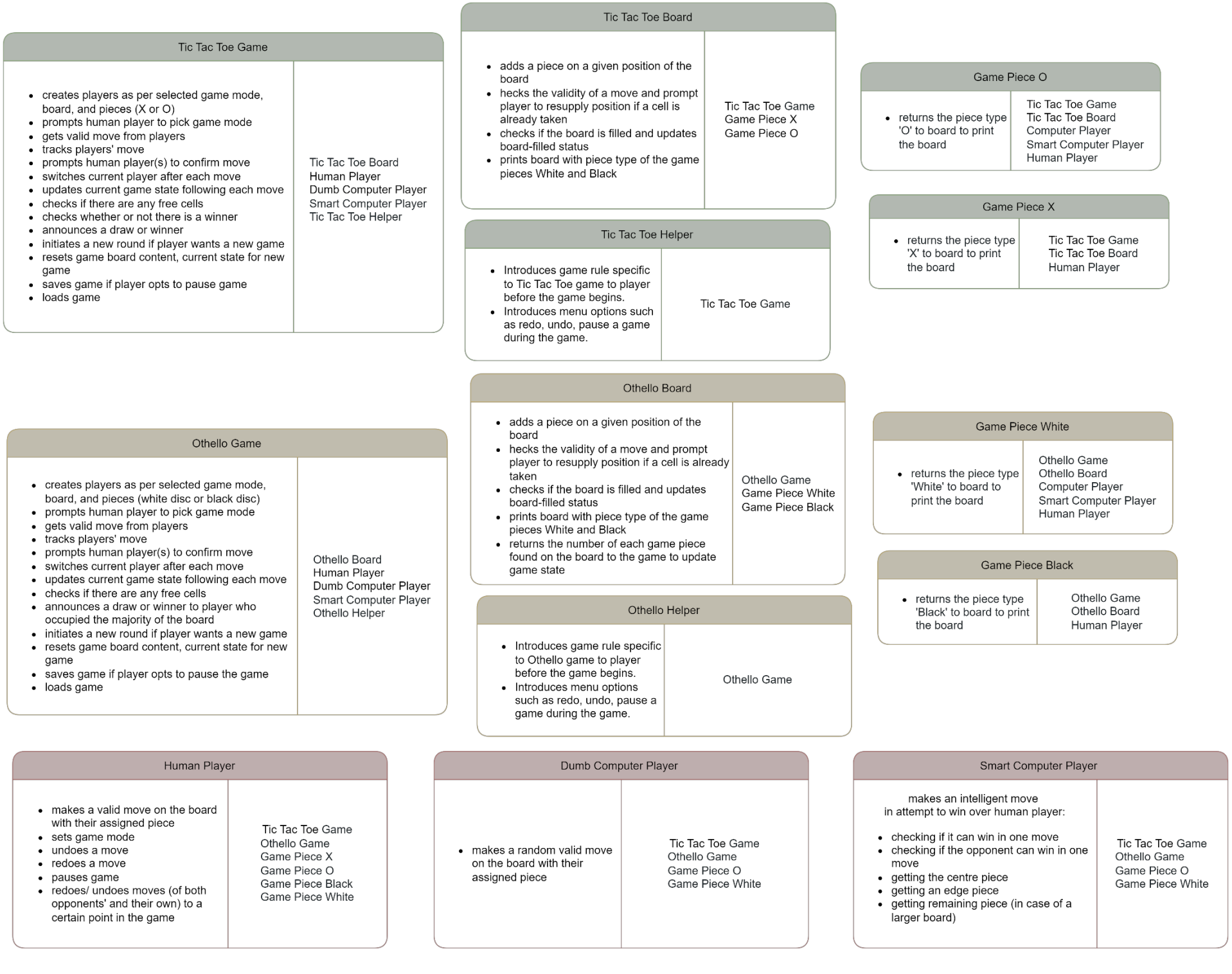
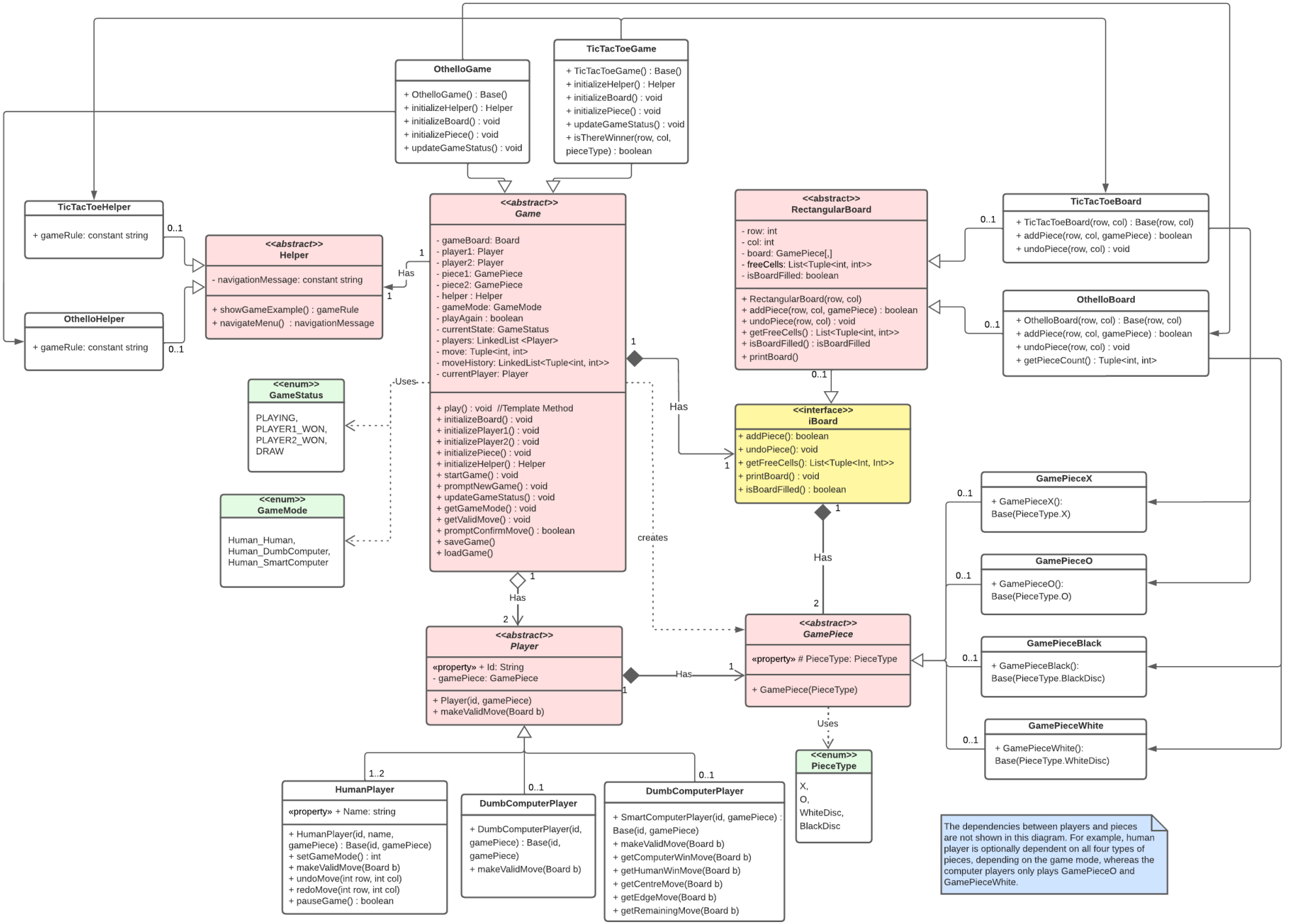
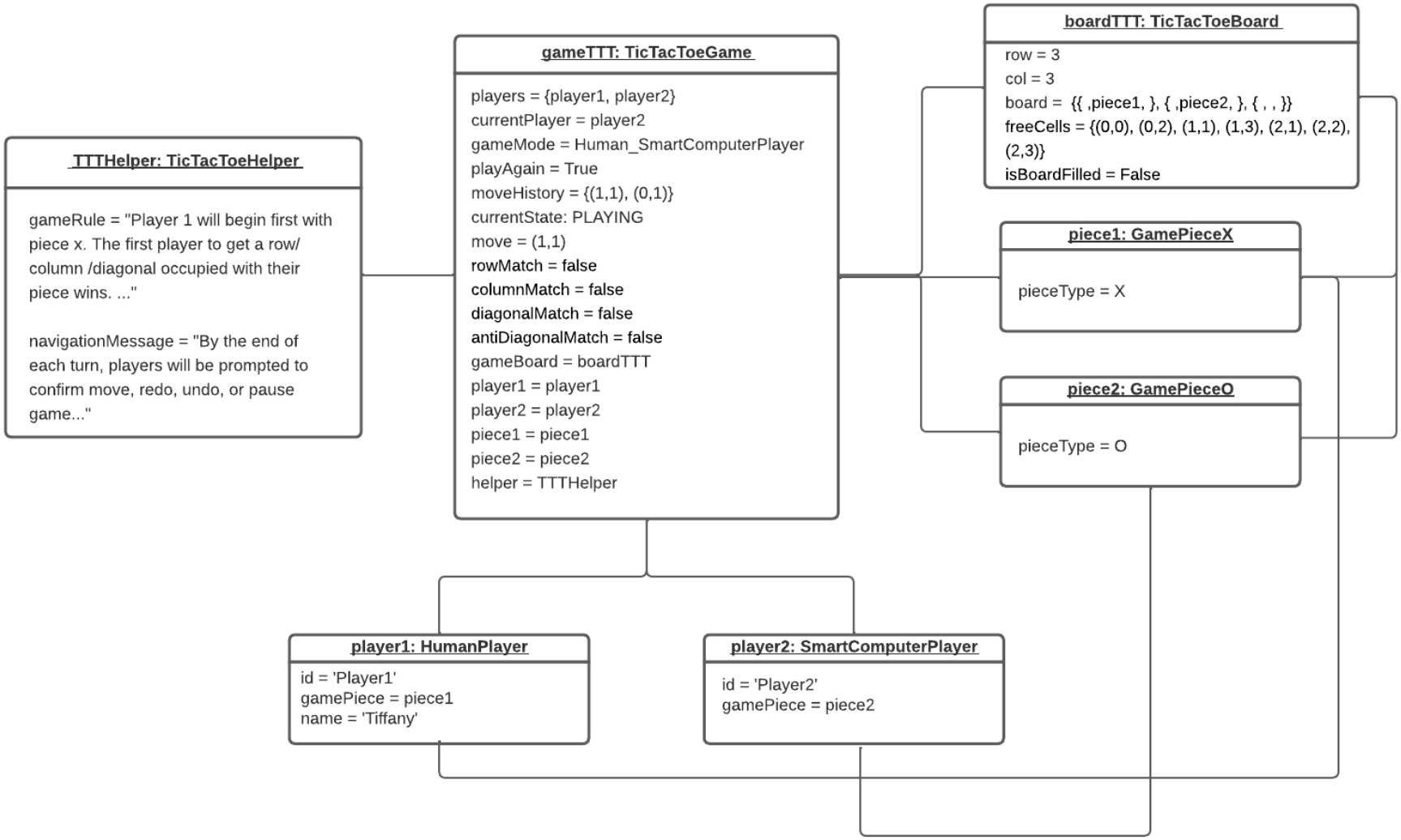
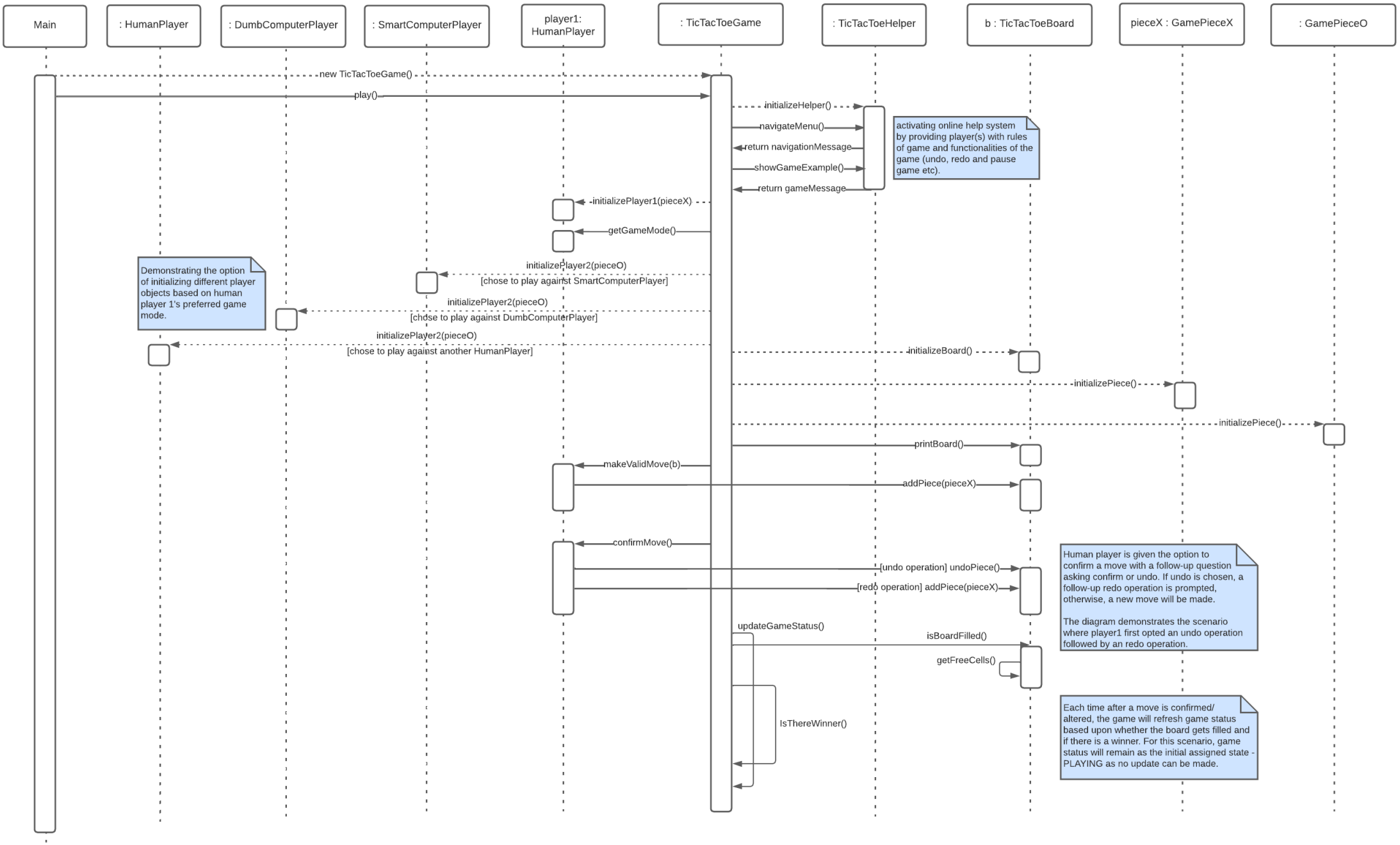
Preliminary Design


Tic Tac Toe Game
creates players as per selected game mode, board, and pieces (X or O)
prompts human player to pick game mode
gets valid move from players
tracks players' move
prompts human player(s) to confirm move
switches current player after each move
updates current game state following each move
checks if there are any free cells
checks whether or not there is a winner
announces a draw or winner
initiates a new round if player wants a new game
resets game board content, current state for new game
saves game if player opts to pause game
loads game
Tic Tac Toe Board
Human Player
Dumb Computer Player
Smart Computer Player
Tic Tac Toe Helper
Tic Tac Toe Board
adds a piece on a given position of the board
hecks the validity of a move and prompt player to resupply position if a cell is already taken
checks if the board is filled and updates board-filled status
prints board with piece type of the game pieces White and Black
Tic Tac Toe Game
Game Piece X
Game Piece O
Game Piece O
returns the piece type 'O' to board to print the board
Tic Tac Toe Game
Tic Tac Toe Board
Computer Player
Smart Computer Player
Human Player
Game Piece X
returns the piece type 'X' to board to print the board
Tic Tac Toe Game
Tic Tac Toe Board
Human Player
Tic Tac Toe Helper
Introduces game rule specific to Tic Tac Toe game to player before the game begins.
Introduces menu options such as redo, undo, pause a game during the game.
Tic Tac Toe Game
Othello Game
creates players as per selected game mode, board, and pieces (white disc or black disc)
prompts human player to pick game mode
gets valid move from players
tracks players' move
prompts human player(s) to confirm move
switches current player after each move
updates current game state following each move
checks if there are any free cells
announces a draw or winner to player who occupied the majority of the board
initiates a new round if player wants a new game
resets game board content, current state for new game
saves game if player opts to pause the game
loads game
Othello Board
Human Player
Dumb Computer Player
Smart Computer Player
Othello Helper
Othello Board
adds a piece on a given position of the board
hecks the validity of a move and prompt player to resupply position if a cell is already taken
checks if the board is filled and updates board-filled status
prints board with piece type of the game pieces White and Black
returns the number of each game piece found on the board to the game to update game state
Othello Game
Game Piece White
Game Piece Black
Game Piece White
returns the piece type 'White' to board to print the board
Othello Game
Othello Board
Computer Player
Smart Computer Player
Human Player
Game Piece Black
returns the piece type 'Black' to board to print the board
Othello Game
Othello Board
Human Player
Othello Helper
Introduces game rule specific to Othello game to player before the game begins.
Introduces menu options such as redo, undo, pause a game during the game.
Othello Game
Human Player
makes a valid move on the board with their assigned piece
sets game mode
undoes a move
redoes a move
pauses game
redoes/ undoes moves (of both opponents' and their own) to a certain point in the game
Tic Tac Toe Game
Othello Game
Game Piece X
Game Piece O
Game Piece Black
Game Piece White
Dumb Computer Player
makes a random valid move on the board with their assigned piece
Tic Tac Toe Game
Othello Game
Game Piece O
Game Piece White
Smart Computer Player
makes an intelligent move in attempt to win over human player:
checking if it can win in one move
checking if the opponent can win in one move
getting the centre piece
getting an edge piece
getting remaining piece (in case of a larger board)
Tic Tac Toe Game
Othello Game
Game Piece O
Game Piece White


OthelloGame
+ OthelloGame() : Base()
+ initializeHelper() : Helper
+ initializeBoard() : void
+ initializePiece() : void
+ updateGameStatus() : void
TicTacToeGame
+ TicTacToeGame() : Base()
+ initializeHelper() : Helper
+ initializeBoard() : void
+ initializePiece() : void
+ updateGameStatus() : void
+ isThereWinner(row, col, pieceType) : boolean
TicTacToeHelper
+ gameRule: constant string
OthelloHelper
+ gameRule: constant string
0..1
<<abstract>>
Helper
- navigationMessage: constant string
+ showGameExample() : gameRule
+ navigateMenu() : navigationMessage
Has
1
<<abstract>>
Game
- gameBoard: Board
- player1: Player
- player2: Player
- piece1: GamePiece
- piece2: GamePiece
- helper : Helper
- gameMode: GameMode
- playAgain : boolean
- currentState: GameStatus
- players: LinkedList <Player>
- move: Tuple<int, int>
- moveHistory: LinkedList<Tuple<int, int>>
- currentPlayer: Player
+ play() : void //Template Method
+ initializeBoard() : void
+ initializePlayer1() : void
+ initializePlayer2() : void
+ initializePiece() : void
+ initializeHelper() : Helper
+ startGame() : void
+ promptNewGame() : void
+ updateGameStatus() : void
+ getGameMode() : void
+ getValidMove() : void
+ promptConfirmMove() : boolean
+ saveGame()
+ loadGame()
Uses
<<enum>>
GameStatus
PLAYING,
PLAYER1_WON,
PLAYER2_WON,
DRAW
<<enum>>
GameMode
Human_Human,
Human_DumbComputer,
Human_SmartComputer
Has
creates
<<abstract>>
RectangularBoard
- row: int
- col: int
- board: GamePiece[,]
- freeCells: List<Tuple<int, int>>
- isBoardFilled: boolean
+ RectangularBoard(row, col)
+ addPiece(row, col, gamePiece) : boolean
+ undoPiece(row, col) : void
+ getFreeCells() : List<Tuple<int, int>>
+ isBoardFilled() : isBoardFilled
+ printBoard()
TicTacToeBoard
+ TicTacToeBoard(row, col) : Base(row, col)
+ addPiece(row, col, gamePiece) : boolean
+ undoPiece(row, col) : void
OthelloBoard
+ OthelloBoard(row, col) : Base(row, col)
+ addPiece(row, col, gamePiece) : boolean
+ undoPiece(row, col) : void
+ getPieceCount() : Tuple<int, int>
<<interface>>
iBoard
+ addPiece(): boolean
+ undoPiece(): void
+ getFreeCells(): List<Tuple<Int, Int>>
+ printBoard() : void
+ isBoardFilled() : boolean
<<abstract>>
Player
«property» + Id: String
- gamePiece: GamePiece
+ Player(id, gamePiece)
+ makeValidMove(Board b)
<<abstract>>
GamePiece
«property» # PieceType: PieceType
+ GamePiece(PieceType)
Uses
<<enum>>
PieceType
X,
O,
WhiteDisc,
BlackDisc
GamePieceX
+ GamePieceX():
Base(PieceType.X)
GamePieceO
+ GamePieceO():
Base(PieceType.O)
GamePieceBlack
+ GamePieceBlack():
Base(PieceType.BlackDisc)
GamePieceWhite
+ GamePieceWhite():
Base(PieceType.WhiteDisc)
HumanPlayer
«property» + Name: string
+ HumanPlayer(id, name, gamePiece) : Base(id, gamePiece)
+ setGameMode() : int
+ makeValidMove(Board b)
+ undoMove(int row, int col)
+ redoMove(int row, int col)
+ pauseGame() : boolean
DumbComputerPlayer
+ DumbComputerPlayer(id, gamePiece) : Base(id, gamePiece)
+ makeValidMove(Board b)
DumbComputerPlayer
+ SmartComputerPlayer(id, gamePiece) : Base(id, gamePiece)
+ makeValidMove(Board b)
+ getComputerWinMove(Board b)
+ getHumanWinMove(Board b)
+ getCentreMove(Board b)
+ getEdgeMove(Board b)
+ getRemainingMove(Board b)
The dependencies between players and pieces are not shown in this diagram. For example, human player is optionally dependent on all four types of pieces, depending on the game mode, whereas the computer players only plays GamePieceO and GamePieceWhite.



Program memory snapshot of a point where the smart computer player has made a move at (1,1) (row2, column2) following human player's initial move at (0,1) (row1, column2).



Sequence Diagram for the Use Case of Human selecting game mode and making an undo and redo operations after the initial move on a 3x3 board in TicTacToe Game

Game rule is such that HumanPlayer1 is always the first one to play and is always the one to play x.