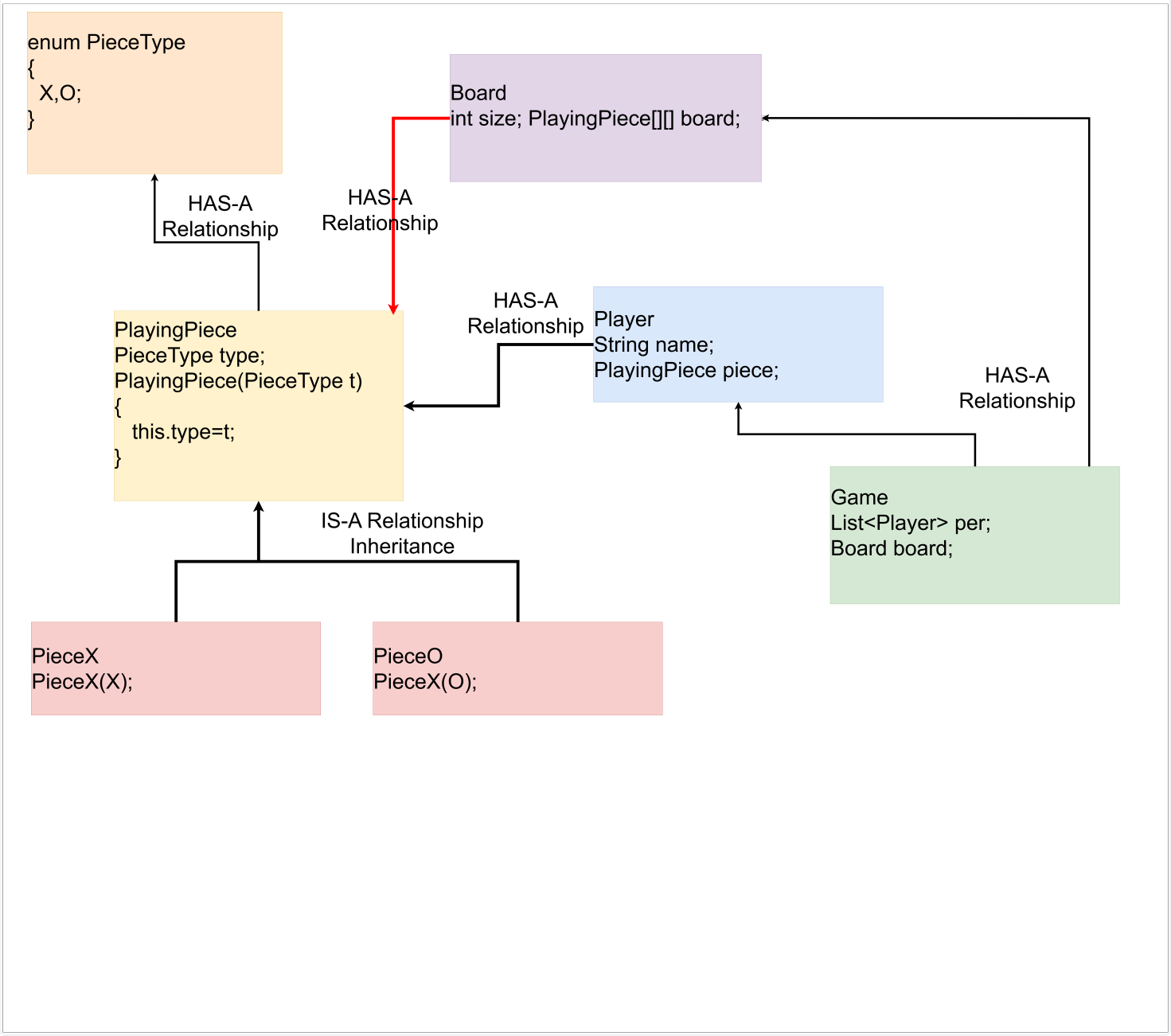
Design Tic Tac Toe game | Tic-Tac-Toe LLD Java | Low Level Design Interview Quest, System Design

Tic Tac Toe game.

x	x	x
x	o	
x		

OBJECTS:

- Piece: X,O,$ #@
- Board n*m
- Players



code:

```
package System_Design.LLD.TIC_TAC_TOE.Model;

public enum PieceType {
    X,O;
}
```

```
package System_Design.LLD.TIC_TAC_TOE.Model;

public class PlayingPiece {
    public PieceType pieceType;
    PlayingPiece(PieceType pieceType){
        this.pieceType=pieceType;
    }
}
```

```
package System_Design.LLD.TIC_TAC_TOE.Model;

public class PlayingPieceO extends PlayingPiece {
    public PlayingPieceO() {
        super(PieceType.O);
    }
}
```

```
package System_Design.LLD.TIC_TAC_TOE.Model;

public class PlayingPieceX extends PlayingPiece {
    public PlayingPieceX() {
        super(PieceType.X);
    }
}
```

```
package System_Design.LLD.TIC_TAC_TOE.Model;

import System_Design.LLD.TIC_TAC_TOE.Pair;

import java.util.ArrayList;
import java.util.List;

public class Board {
    public int size;
    public PlayingPiece[][] board;

    public Board(int size){
        this.size = size;
        board=new PlayingPiece[size][size];
    }

    public boolean addPiece(int row, int col,PlayingPiece playingPiece){
        if(board[row][col]!=null){
            return false;
```

```
        }
        board[row][col] = playingPiece;
        return true;
    }

    public List<Pair<Integer,Integer>> getFreeCells(){
        List<Pair<Integer,Integer>> freeCells = new ArrayList<>();

        for (int i=0;i<size;i++){
            for(int j=0;j<size;j++){
                if (board[i][j]==null){
                    Pair<Integer,Integer> rowColums = new Pair<>(i,j);

                    freeCells.add(rowColums);
                }
            }
        }
        return freeCells;
    }

    public void printBoard() {

        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                if (board[i][j] != null) {
                    System.out.print(board[i][j].pieceType.name() + "   ");
                } else {
                    System.out.print("    ");

                }
                System.out.print(" | ");
            }
            System.out.println();

        }
    }

    }
```

```
package System_Design.LLD.TIC_TAC_TOE.Model;

public class Player {
    public String name;
    public PlayingPiece playingPiece;

    public Player(String name,PlayingPiece piece){
        playingPiece=piece;
        this.name=name;
```

```
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public PlayingPiece getPlayingPiece() {
        return playingPiece;
    }

    public void setPlayingPiece(PlayingPiece playingPiece) {
        this.playingPiece = playingPiece;
    }
}
```

```
package System_Design.LLD.TIC_TAC_TOE;

public class Pair<T,U> {
    private final T key;
    private final U value;

    public Pair(T key, U value) {
        this.key = key;
        this.value = value;
    }

    public T getKey() {
        return this.key;
    }

    public U getValue() {
        return this.value;
    }
}
```

```
package System_Design.LLD.TIC_TAC_TOE;

import System_Design.LLD.TIC_TAC_TOE.Model.*;

import java.util.Deque;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;
```

```
public class TicTacToeGame {

    Deque<Player> players;
    Board gameBoard;

    TicTacToeGame(){
        initializeGame();
    }

    private void initializeGame() {
        gameBoard = new Board(3);
        players = new LinkedList<>();
        PlayingPieceX pieceX = new PlayingPieceX();
        Player player1 = new Player("Player1",pieceX);

        PlayingPieceO pieceO = new PlayingPieceO();
        Player player2  = new Player("Player2",pieceO);

        players.add(player1);
        players.add(player2);

    }

    public String startGame() {
        boolean noWinner=true;
        while (noWinner){

            //take out the player whose turn is and also put the player in the list back
            Player playerTurn = players.removeFirst();


            //get the free space from the board
            gameBoard.printBoard();

            List<Pair<Integer,Integer>> freeSpaces = gameBoard.getFreeCells();

            if (freeSpaces.isEmpty()){
                noWinner=false;
                continue;
            }

            //read the user Input
            System.out.println("Player; "+playerTurn.name+" Enter row,column: ");
            Scanner sc = new Scanner(System.in);
            String s = sc.nextLine();
            String[] values = s.split(",");
            int row = Integer.valueOf(values[0]);
            int col = Integer.valueOf(values[1]);
```

```
            //place the piece
            boolean pieceAddedSuccessFully = gameBoard.addPiece(row,col,
playerTurn.playingPiece);
            if (!pieceAddedSuccessFully){
                //player can not inser the piece into the cell, player has to choose
another cell
                System.out.println("Incorect position coosen, try againg");
                players.addFirst(playerTurn);
                continue;
            }

            players.addLast(playerTurn);
            boolean winner = isThereWinner(row,col,playerTurn.playingPiece.pieceType);
            if (winner){
                return playerTurn.name;
            }

        }
        return "tie";
    }

    private boolean isThereWinner(int row, int col, PieceType pieceType) {
        //we can also apply n -queen problem
        boolean rowMatch = true;
        boolean colMatch = true;
        boolean diagonalMatch = true;
        boolean antiDiagonalMatch = true;

        for (int i=0;i<gameBoard.size;i++){
            if (gameBoard.board[row][i]==null||gameBoard.board[row]
[i].pieceType!=pieceType){
                rowMatch=false;
            }
        }

        for (int i=0;i<gameBoard.size;i++){
            if (gameBoard.board[i][col]==null||gameBoard.board[i]
[col].pieceType!=pieceType){
                colMatch=false;
            }
        }

        for (int i=0,j=0;i<gameBoard.size;i++,j++){
            if (gameBoard.board[i][j]==null||gameBoard.board[i][j].pieceType!=pieceType)
{
                diagonalMatch=false;
            }
        }
```

```
        for (int i=0,j=gameBoard.size-1;i<gameBoard.size;i++,j--){
            if (gameBoard.board[i][j]==null||gameBoard.board[i][j].pieceType!=pieceType)
{
                antiDiagonalMatch=false;
            }
        }

        return rowMatch||colMatch||diagonalMatch||antiDiagonalMatch;


    }
}
```

```
package System_Design.LLD.TIC_TAC_TOE;

public class MainGame {
    public static void main(String[] args){
        TicTacToeGame ticTacToeGame = new TicTacToeGame();
        System.out.println("game winner is: "+ticTacToeGame.startGame());
    }
}
output:
     |      |      |
     |      |      |
     |      |      |
Player; Player1 Enter row,column:
1,2
     |      |      |
     |      | X    |
     |      |      |
Player; Player2 Enter row,column:
1,2
Incorect position coosen, try againg
     |      |      |
     |      | X    |
     |      |      |
Player; Player2 Enter row,column:
1,1
     |      |      |
     | O    | X    |
     |      |      |
Player; Player1 Enter row,column:
1,2
Incorect position coosen, try againg
     |      |      |
     | O    | X    |
     |      |      |
Player; Player1 Enter row,column:
```

```
2,0
     |      |      |
     | O    | X    |
X    |      |      |
Player; Player2 Enter row,column:
2,1
     |      |      |
     | O    | X    |
X    | O    |      |
Player; Player1 Enter row,column:
0,2
     |      | X    |
     | O    | X    |
X    | O    |      |
Player; Player2 Enter row,column:
0,1
game winner is: Player2

Process finished with exit code 0
```