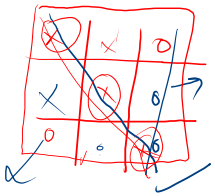
Design Tic Tac Toe

2 - Player

3 x 3 → board



Requirements

↳ Players

↳ Board

Main class

Game class

Player class → name → String

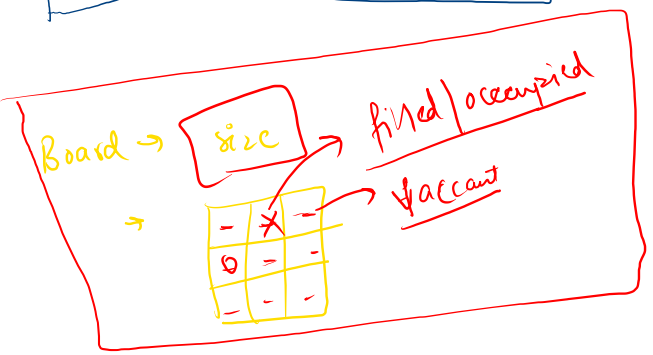
symbol → char

Ram

X , O , #, !

Board → size → filled / occupied

→ vaccant



arr → int → 3 → 2D

int arr[ ][ ];

```java
public static void main(String[] args) {
    Player p1 = new Player();
    p1.setName(name: "A");
    p1.setSymbol(symbol: '0');

    Player p2 = new Player();
    p2.setName(name: "B");
    p2.setSymbol(symbol: 'X');


}
```

Player → name → string
↳ symbol → char

2 Players

3x3 board

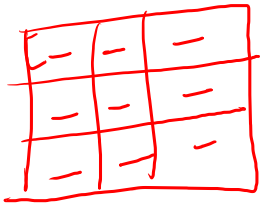
```java
Board board = new Board(size: 3);
```

↳ size = 3

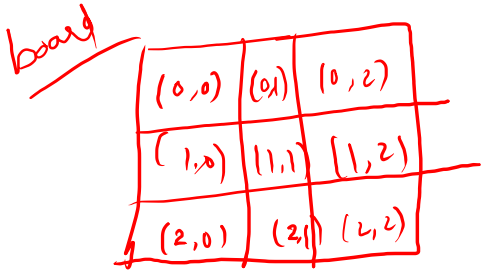
A → 0

B → X

board



```java
public class Game {

    Player [] players;
    Board board;
    int turn;
    int noOfMoves;
    boolean gameOver;
    String zero;
    String cross;

    public Game(Player [] players, Board board){

        this.players=players;
        this.board=board;
        this.turn=0;
        this.noOfMoves=0;
        this.gameOver=false;

        StringBuilder z = new StringBuilder();
        StringBuilder c = new StringBuilder();

        for(int i=0;i<board.size;i++){
            z.append(c: '0');
            c.append(c: 'X');
        }
        zero = z.toString();
        cross = c.toString();

    }

    public void printBoard(){
        for(int i=0;i<board.size;i++){
            for(int j=0;j<board.size;j++){
                System.out.print(board.board[i][j]+" ");
            }
            System.out.println();
        }
    }

}
```

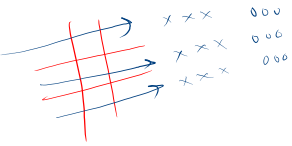


1-9

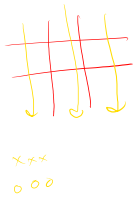
2

```java
public void play(){
    printBoard();
    int n=board.size;

    while(!gameOver){
        noOfMoves++;
        int idx = getIndex();

        int row =idx/n;
        int col = idx%n;

        board.board[row][col]=players[turn].getSymbol();
        if(noOfMoves >= n*n) {
            System.out.println(x: "Game Draw");
            return;

        }

        if(noOfMoves >=  2*n-1 && checkCombinations() == true){
            gameOver = true;
            System.out.println("Winner is : "+ players[turn].getName()
            return ;
        }

        turn =(turn +1)%n;

    }
}
```

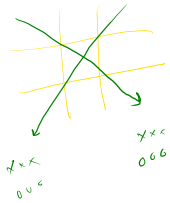
```java
public int getIndex(){
    while(true){

        System.out.println("Player: "+ players[turn].getName() +" give one position");
        Scanner scn = new Scanner(System.in);

        int pos= scn.nextInt()-1;
        int n=board.size;
        int row=pos/n, col=pos%n;

        if(row<0 || col <0 || row>=n || col >=n){
            System.out.println(x: "Invalid position");
            continue;
        }

        if(board.board[row][col]!='-'){
            System.out.println(x: "Position already occupied");
            continue;
        }
        return pos;

    }

}
```



⇒ Row wise



⇒ Column wise



Diagonal

n = 4

	0	1	2	3
0	0	-1	-2	-3
1	1	0	-1	-2
2	2	1	0	-1
3	3	2	1	0

⇒ i - j <= 0

	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

i + j == n-1

```java
public boolean checkCombinations(){

    int n=board.size;

    //  Row wise
    for(int i=0;i<n;i++){
        StringBuilder sb =new StringBuilder();
        for(int j=0;j<n;j++){
            sb.append(board.board[i][j]);
        }
        String pattern =sb.toString();

        if(pattern.equals(zero) || pattern.equals(cross)){
            return true;
        }
    }
```

```java
// ColumnWise

for(int i=0;i<n;i++){
    StringBuilder sb= new StringBuilder();
    for(int j=0;j<n;j++){
        sb.append(board.board[j][i]);
    }

    String pattern =sb.toString();

    if(pattern.equals(zero)  || pattern.equals(cross)){
        return true;
    }
}
```

```java
// Diagonal

StringBuilder sb= new StringBuilder();

int i=0,j=0;
while(i<n){
    sb.append(board.board[i][j]);
    i++;
    j++;
}


String pattern =sb.toString();

if(pattern.equals(zero) || pattern.equals(cross)){
    return true;
}
```

```java
    // Anti Diagonal
    sb= new StringBuilder();

     i=0;
     j=n-1;
    while(i<n){
        sb.append(board.board[i][j]);
        i++;
        j--;
    }

     pattern =sb.toString();

    if(pattern.equals(zero) || pattern.equals(cross)){
        return true;
    }

    return false;

}
```