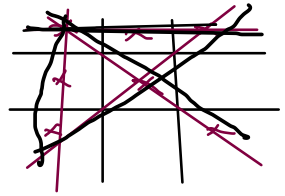
PROJECT-01

TIC TAC TOE

[X] [O]

players
↳ p1
↳ p2



→ 3x3

→ Rowise
→ Columnwise
→ Diagonalwise
→ Anti diagonalwise

size = 3

3 $\boxed{9 == 3 \times 3}$

O	O	O
X		X
	X	

p1 → Ram → X
p2 → Ravi → O

X	X	O
O	X	X
X	O	O

⇒ Draw

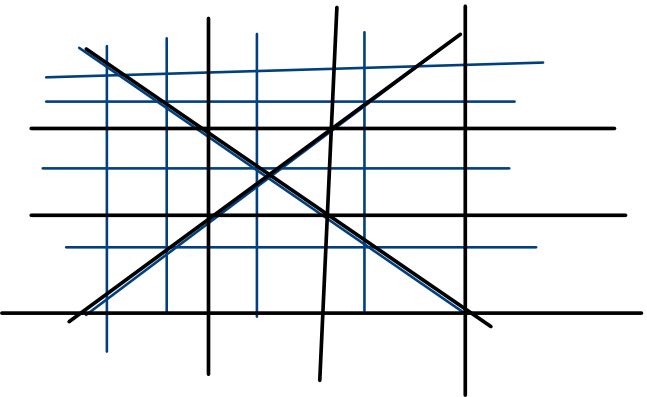
O		
O	X	X
O		

↳ Ravi won

p1 → Ram → X
p2 → Ravi → O

X	X	X
	O	
O		

→ Ram won



↳ Design Tic Tac Toe

↳ Requirements → 1 board → define → size of board ($n \times n$)

3 × 3

4 × 4

5 × 5

→ 2 Players (A, B)

→ Rules

↳ If any row, column, diagonal, anti diagonal is filled with same symbol

↳ considered as winner

X	X	X
O	X	O
X	X	X

→ Outcomes

↳ Player 1 → winner

↳ Player 2 → winner

↳ Draw

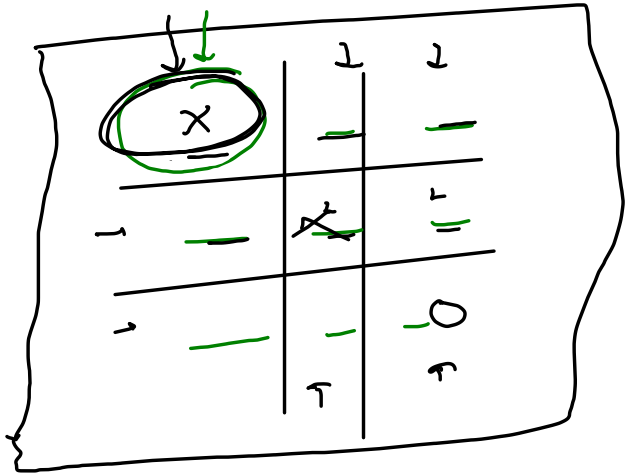
→ Board (5 mins)

↳ Board class

↳ int size → user

↳ character matrix

↳ default character



p1 →

if (=

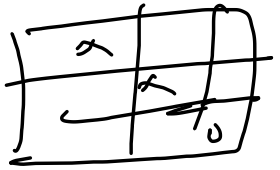
p2

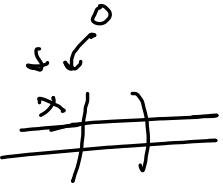
if (→

↳ fn → default characters

↳ constructor

↳ fn. printBoard Config





→ if(default character)

→ Player

→ player class

(5-6) mins

↳ ① name of the player

↳ ② age →

↳ ③ address of the player

↳ ④ player contact number

↳ ⑤ player email id

↳ ⑥ player symbol

Ravi

23

UP

x — —

o —

↳ initialise (constructor)

↳ fn → set

↳ setting the properties of a player

↳ fn → get → name, age, address, contact, email id, symbol

↳ purpose is to return the details of a player.

↳ set → name, symbol, contact, email id

↳ set → name, symbol

fn → get → name, age, symbol

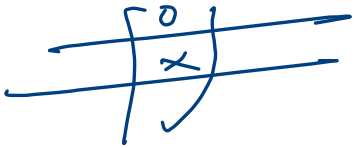
fn → get → name, symbol

↳ validation of player details

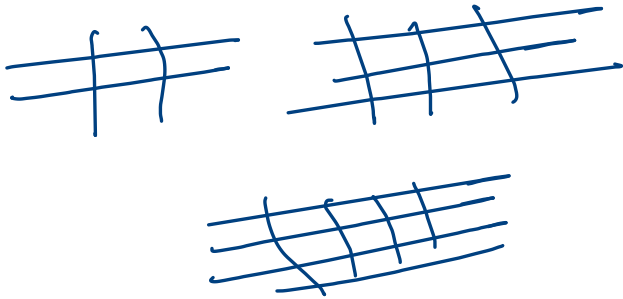
→ Game rules

↳ Game class

↳ Player details → (name, symbol)



↳ Board →

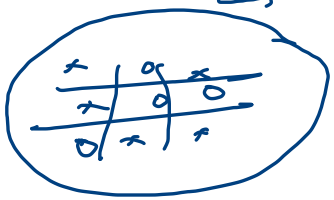


↳ turn → (allow us to know which player turn is it)

↳ no of Moves →

↳ gameOver

↳



9 = 3 * 3

zeros = "000"
cross = "xxx"
size = 3

zeros = "0000"
cross = "xxxx"

↳ constructor (for initialization)

↳ print Board Config

size = 3

size → 1-26

	0	1	2
0	(0,0) 1	(0,1) 2	(0,2) 3
1	(1,0) 4	(1,1) 5	(1,2) 6
2	(2,0) 7	(2,1) 8	(2,2) 9

→ select the position

↳ valid (1 - 9) → vaccant → no issue (place the symbol)

→ Occupied → please enter the position again

↳ invalid (< 0 or > 9)

↳ please enter position again.

[1 - 9] → 6

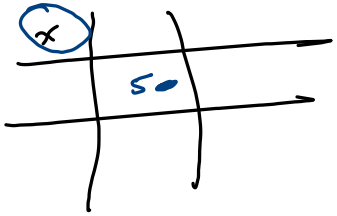
0 - 8

↓

5

row

col



row = 5/3 = 1

col = 5%3 = 2

(1,2)

8 → 7 → row = 7/3 = 2

col = 7%3 = 1

(2,1)

→ position

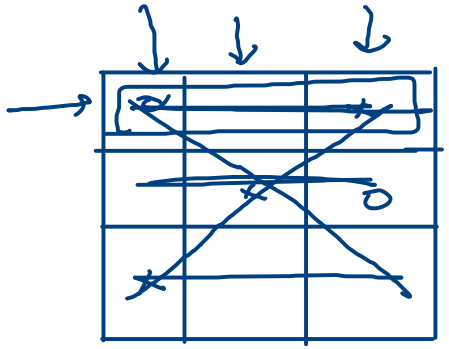
↳ valid → vaccant → place/fill the symbol

→ occupied → please enter the position again

↳ invalid → enter the pos again

→ valid position

5 zeros, cross



→ Rowise

```
for(int i=0; i< sz; i++) {
    string bw a
    for(int j=0; j<sz; j++) {
        a.append(matrix[i][j])
```

(a == zeros || a == cross)

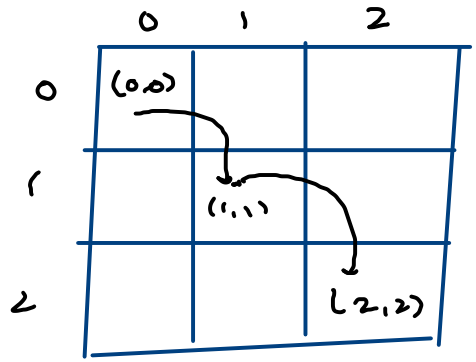
return true;

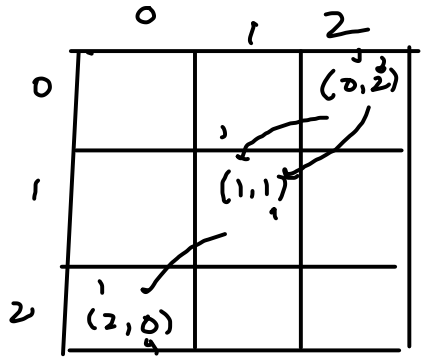
000

xxx

row = ++

col = ++





→ row = ++

col = --

Rowise

	0	1	2
0			
1	x	O	x
2			

	0	1	2
0	x		
1	x		
2	x		

pattern = "xxx"

	0	1	2
0	x		
1		O	
2			O

→ pattern = "x O O"

Sb. = "xOx"

pattern = "x O x"

zero = "OOO"

cross = "xxx"

	0	1	2
0			x
1		x	
2	O		

→ pattern = "x x O"

```java
public boolean checkCombinations() {
    int sz = board.size;

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

        String pattern = sb.toString();
        if (pattern.equals(zero) || pattern.equals(cross)) {
            return true;
        }
    }

    // Columwise
    for(int i=0;i<sz;i++) {
        StringBuilder sb = new StringBuilder();
        for(int j=0;j<sz;j++) {
            sb.append(board.matrix[j][i]);
        }
        String pattern = sb.toString();
        if (pattern.equals(zero) || pattern.equals(cross)) {
            return true;
        }
    }

    // Diagonal

    int i=0,j=0;
    StringBuilder sb = new StringBuilder();

    while (i<sz) {
        sb.append(board.matrix[i][j]);
        i++;
        j++;
    }

    String pattern = sb.toString();
    if (pattern.equals(zero) || pattern.equals(cross)) {
        return true;
    }
```

```java
    // Anti Diagonal

    i=0;
    j=sz-1;
    sb = new StringBuilder();

    while (i<sz) {
        sb.append(board.matrix[i][j]);
        i++;
        j--;
    }

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

    return false;

}
```

A → turn → 0 → X

B → turn → 1 → O

idx = 5

row = 1

col = 2

3 2

2×3 2-1

no of Move = 1

0

X		O
	X	
X		

2×3-1 ⇒ 5

board.matrix[1][2] = players[turn].getSymbol → combination

→ idx → 4