LLD

Design Snakes and Ladders

Only

Req Gathering

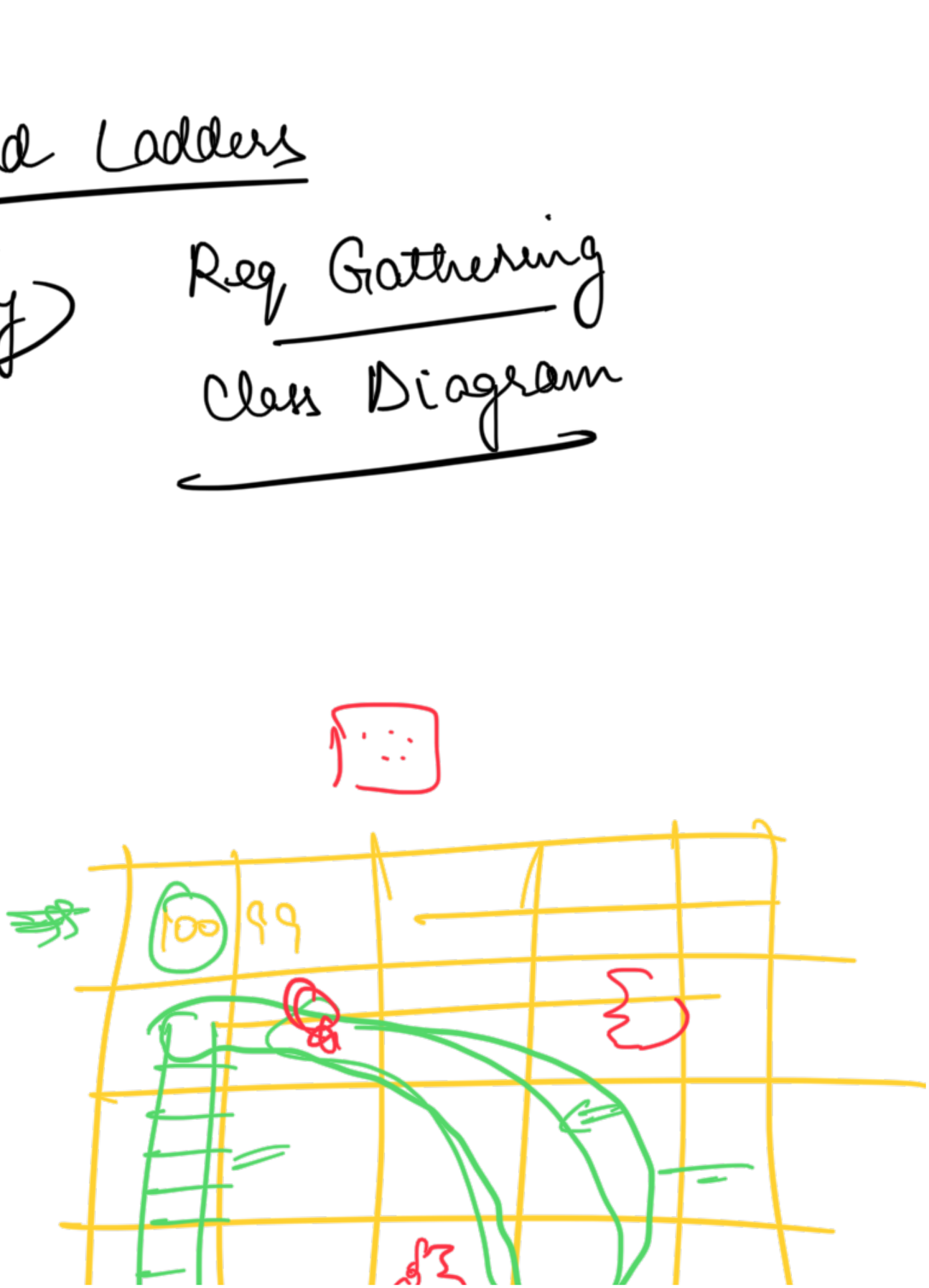
Class Diagram

Overview ⇒ ✓

Gathering Req ⇒ ✓

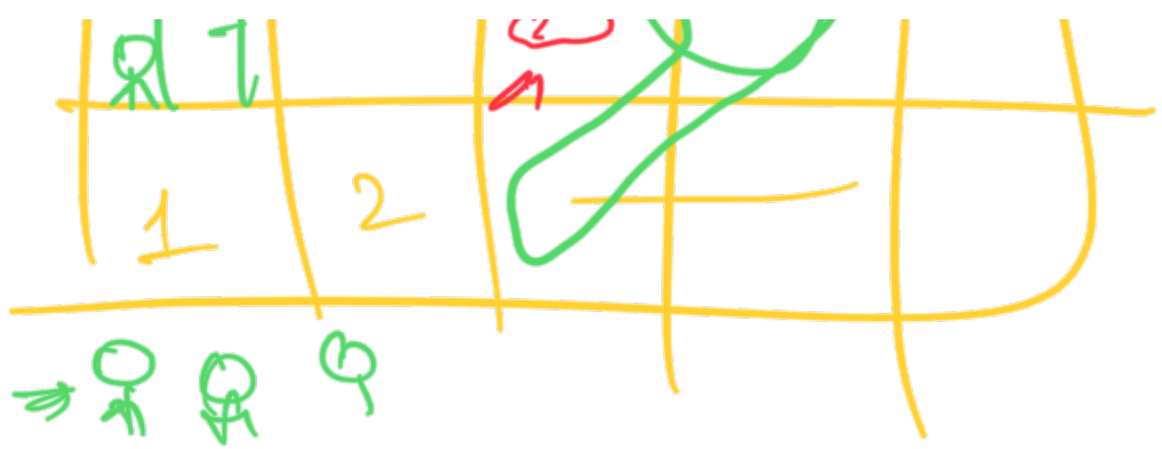
→ Game can be dfn
any # of players (>=2)

→ Only considering Human
players



player.

→ Board can be of any varying size decided by the client.



⇒ Pos^n of Snakes and ladders are random and decided automatically at the start of the game

⇒ # of Snakes and ladders is also random

→ the Snakes & ladders are at valid pos^n

→ they dont conflict with each other

⇒ Pos^n of Snakes and ladders is also random

⇒ Size of Snakes

⇒ In future diff other types of entities can also come up ⇒ eg frogs - taking from one place to other

→ A player enters the game at only pos 1 only after they get a 1 or 6 =

⇒ Only one dice in a game.

⇒ The max # of dice is configu

→ Current Scope

→ future Scope

9

1 to 4

6

6 - 6

⇒ 0

1 to 6

⇒ The same player should take a move after if they get [max No]

⇒ Game ends when all but one have reached 100.

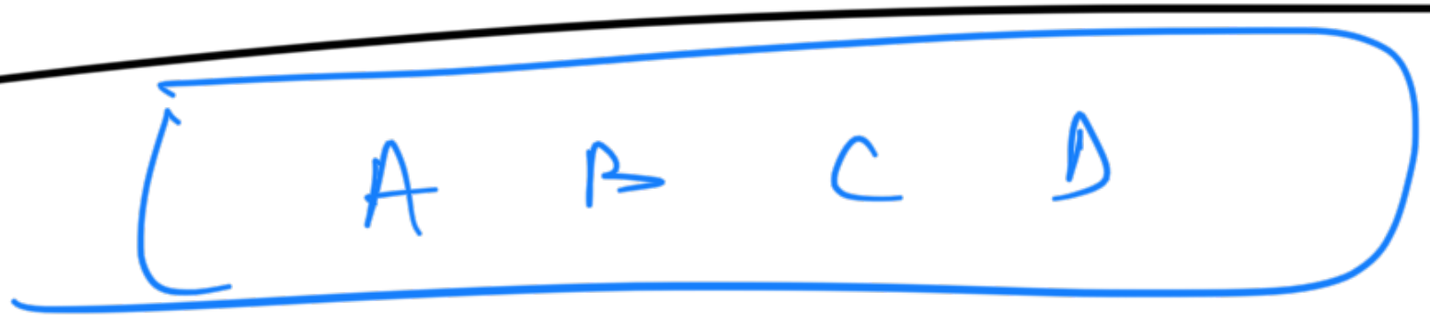
⇒ Winner is someone who reached 100 all buttons have size of one box

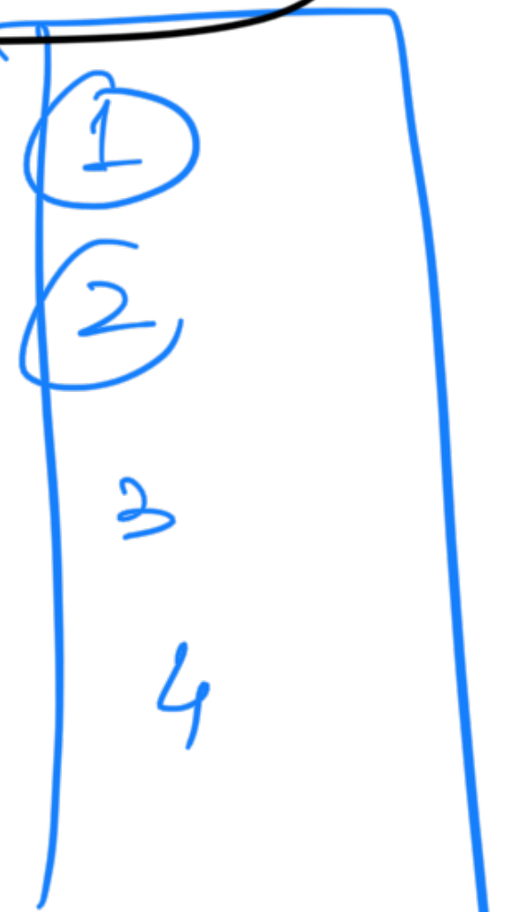
⇒ Can't Skip

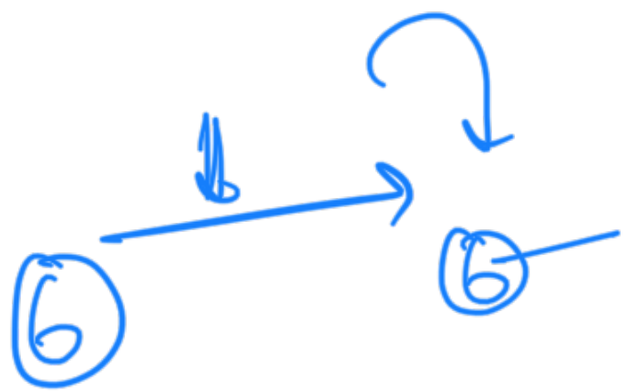
99 → 2 ⇒ Invalid

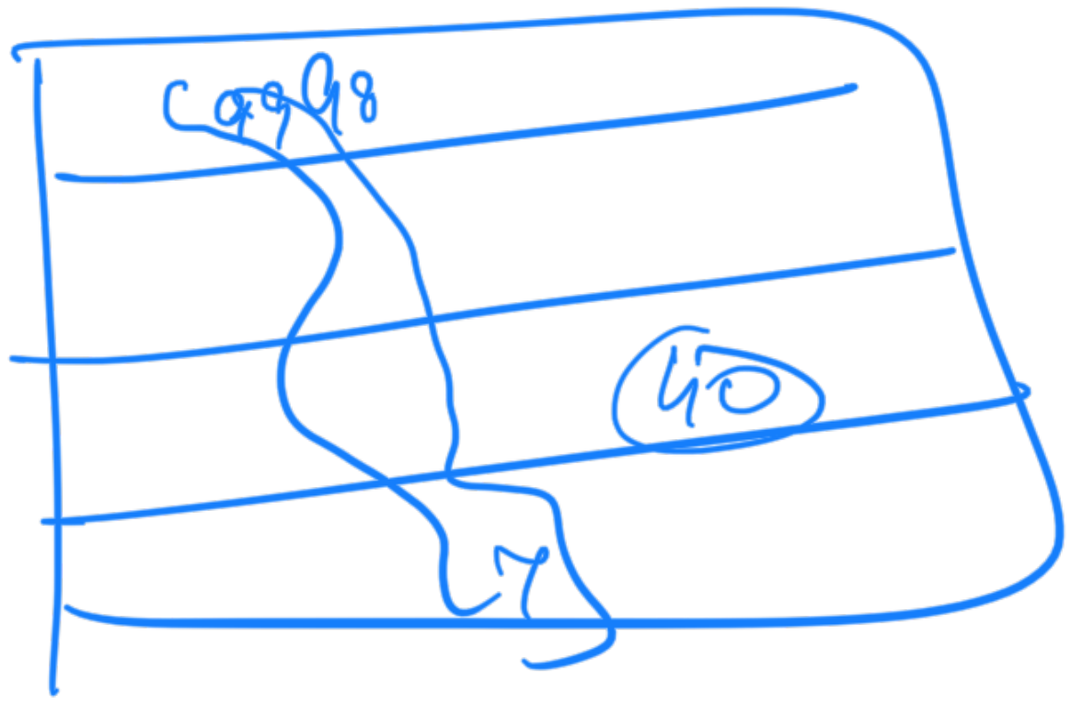
⇒ For every game maintain the leaderboard

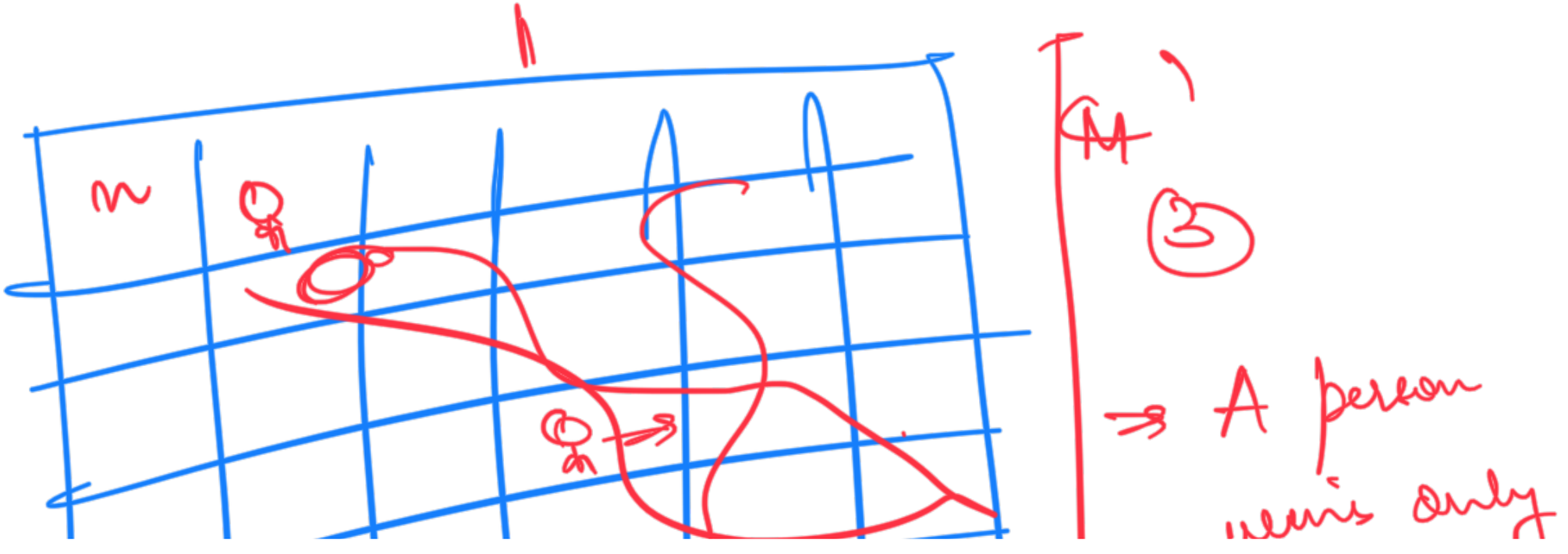
⇒ There can be multiple buttons per player

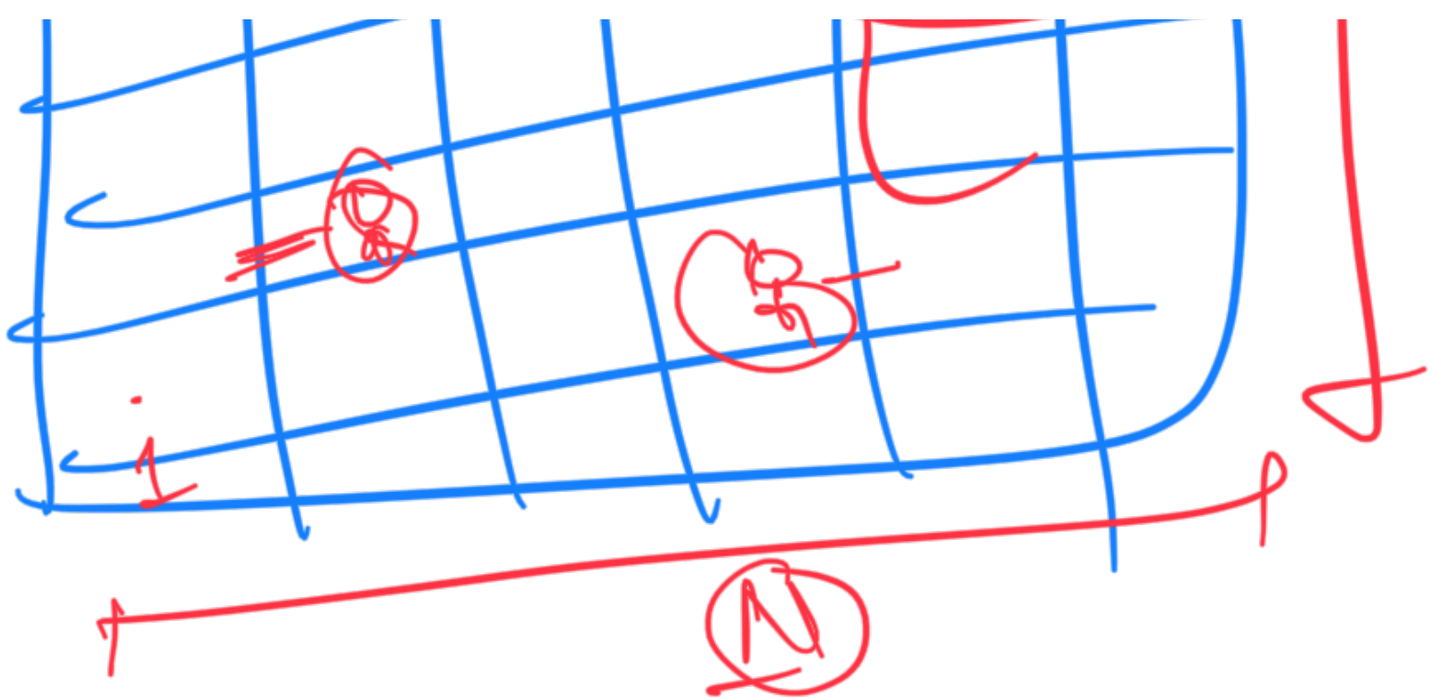












when everyone reaches 100

=

→ Allow a color per row

100

200

Gam

100

1

Game

Board

Dice

List<Player>

max Button per turn

game Status → IN_PROGRESS / COMPLETED

rankings → List<Player>

last Player Idx

move Strategy

1

2

Board

int Size

Map<Int, ForeignEntity>

key : pos

value : ?

Ladder → to → from

Snake from > to

to

from

ForeignEntity

Type

int from

int to

SNAKE

LADDER

FROG

FE(Type)

Snake

Snake (from, to)

Ladder

Frog

Prog

from

Deathen

to

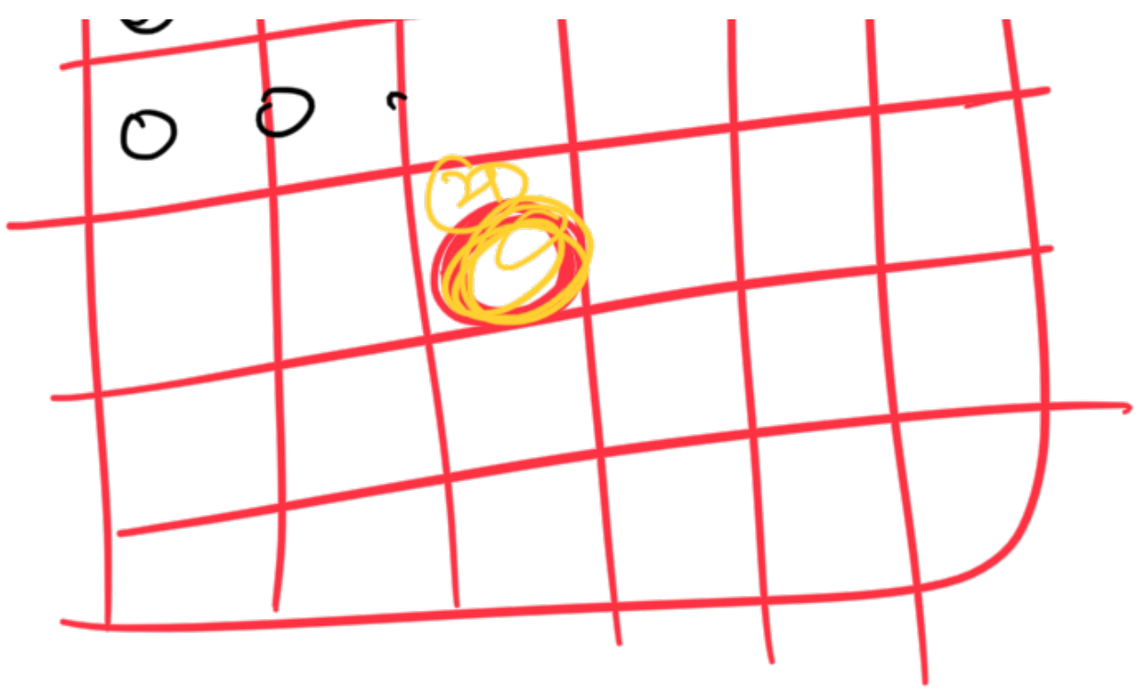
13 → 17

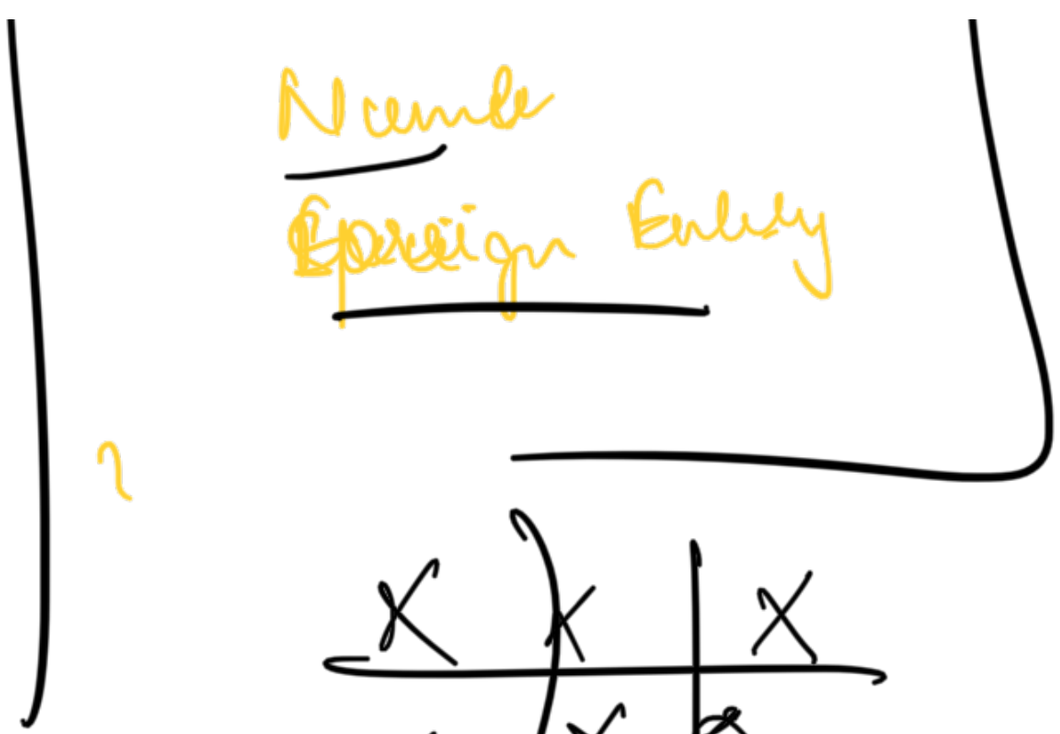
21 → 27

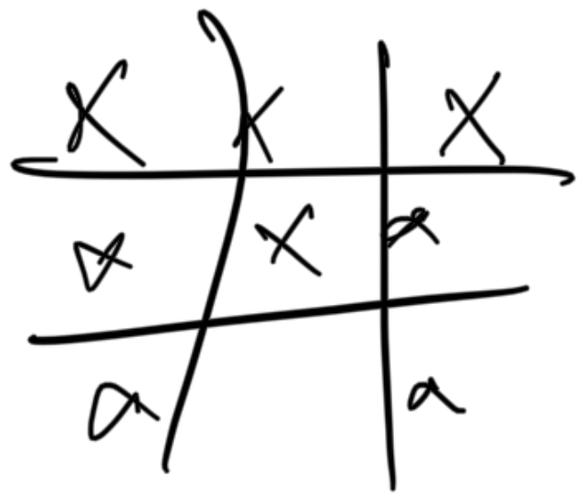
100

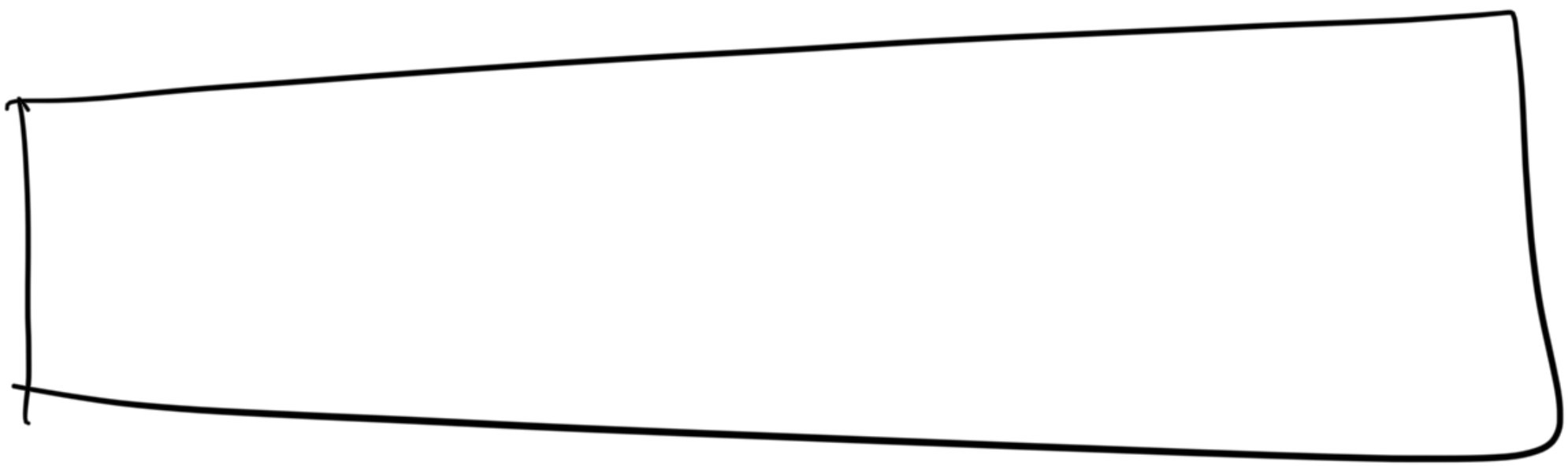
100

Cell C




Ncmb
Foreign Entity





make Wide ( ) {

Dice = 5

}

Dice

-maxNo

- roll() {

1 — maxNo

}

Button

Player

List<Button>

PlayerIA

'Naman'

- position
- Status
- Name

LOCKED

IN_GAME

ENDED

Naman_PI

Naman-PL

P1-1

P1-1

P1-2

<<Button Starting Strategy>

can start (value)

return value == 1 or
value == -1

makeMove() {

if (button.status() == LOCKED)

if (bStartStrategy.canStart(value)

button.status = NO-GAME

button.pos = 1

}

b.o

97 → 100

<< Player Move Strategy >>

move (Button → value, Board)

Normal Player Move Strategy

// Check if it doesn't overflow
// handle Snakes and ladder
// update the status of the button

TTT

Game

Board

Dice

List <Player>

total ButtonPerPlayer

Game Status

List <Player> ranking

lastPlayer MoveIndex

Make Move Strategy

Board

int dimension

Map<int, Foreign Entity>

Foreign Entity

Type

from

to

[Can Start Strategy]

Dice

- maxNo

- roll()

Ladder

Snake

Prop

Next Prime

Player

id

List <Button>

color

Button

position

Status → LOCKED
Status → IN-GAME
Status → COMPLETED

<< HandleMoveStrategy >>

move (Button, value, Board)

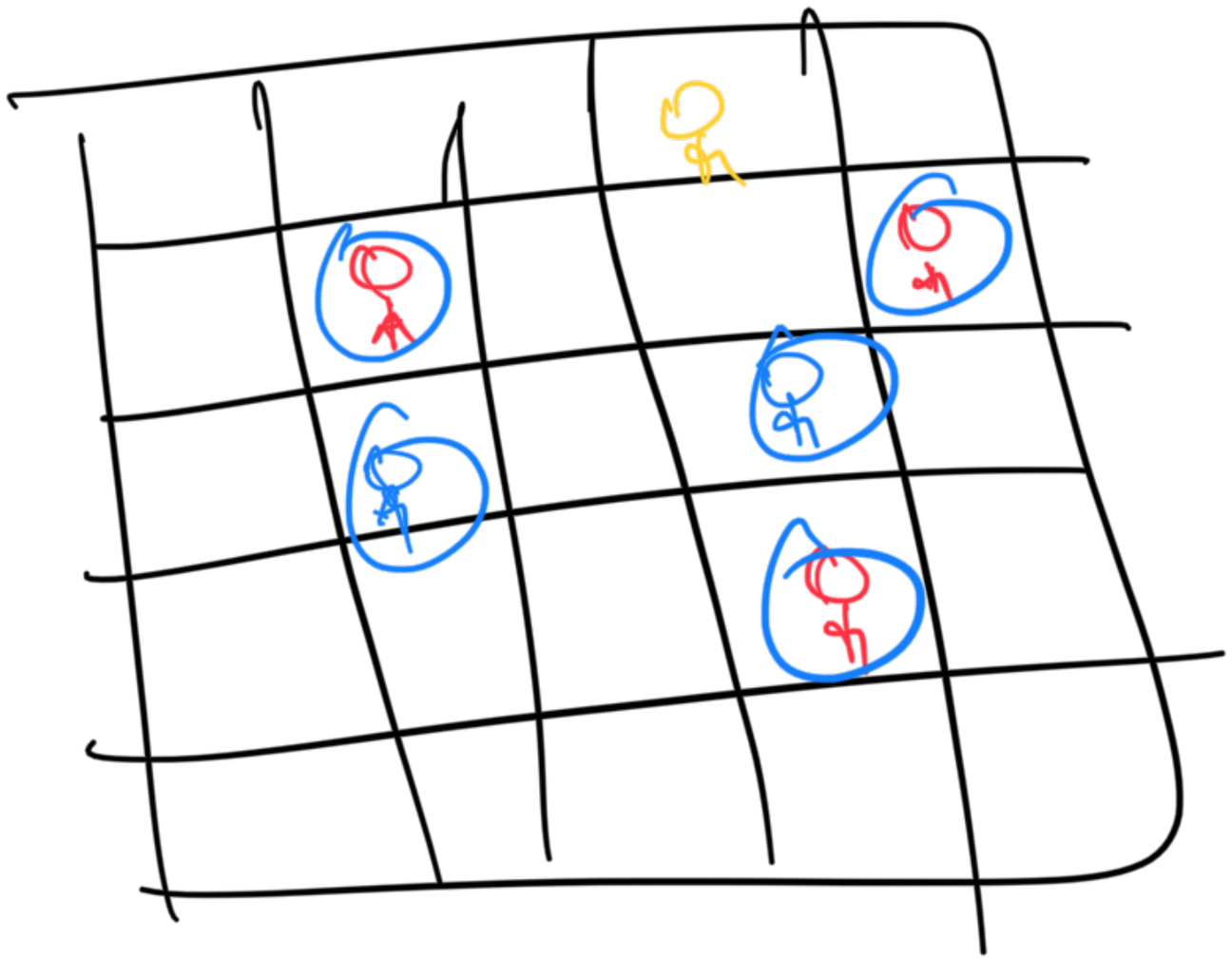
// execution of movement will happen here

<< UnlockDiceStrategy >>

canUnlock (value)

Next Sunday

→ My code



Game

low level design

Thurs / Fri

Game Simulator

HOMEWORK

① Completet TTT

Game Controller

↓

Game

(UI, Code)

② Watch & bL
Complementty implemen
via learnings of
TTT

③ undo() in C bL

④ 6-656 go back to
pre de

31 st July Code