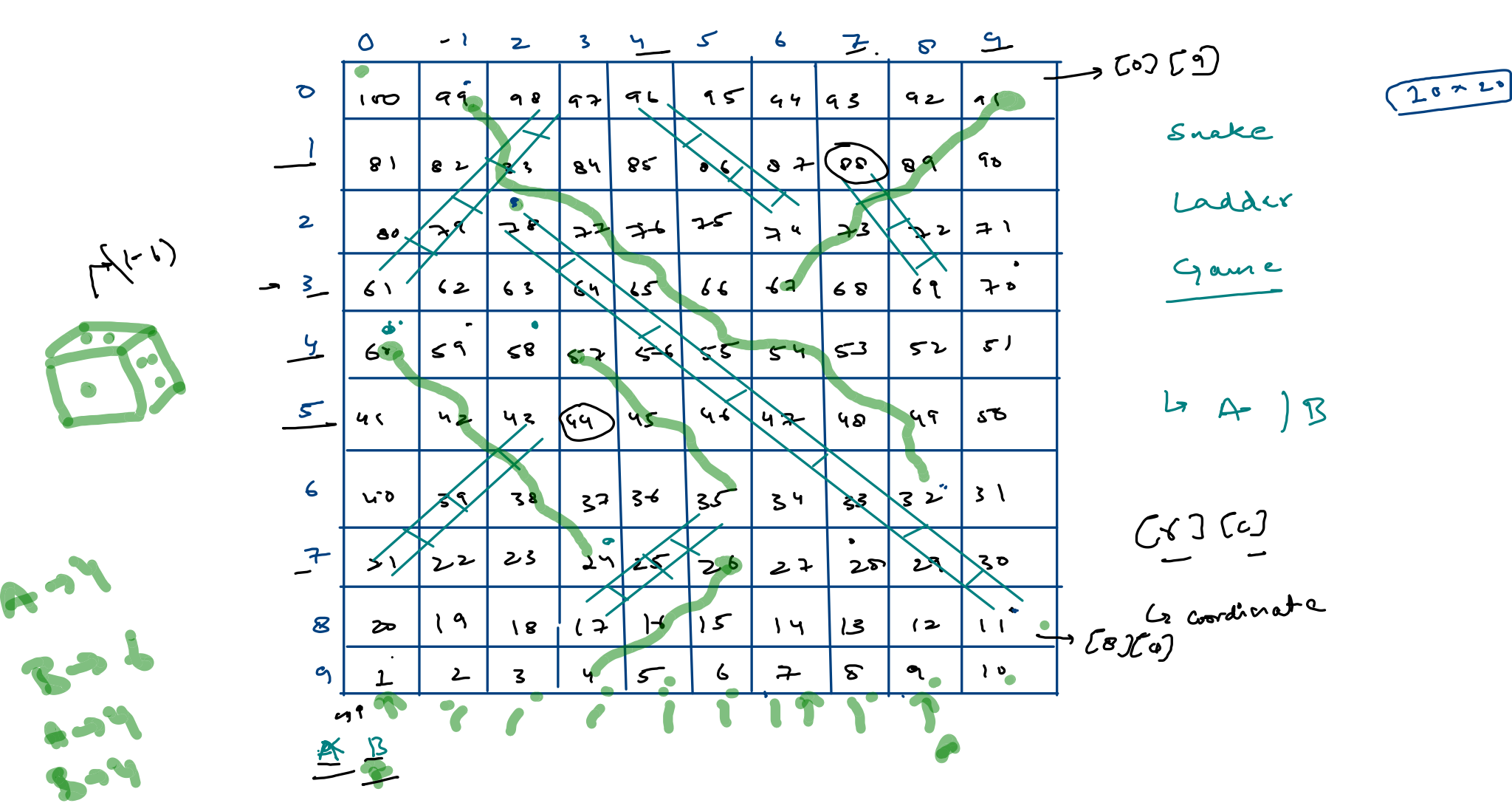
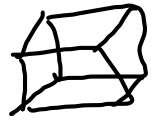
Snake Ladder Game


Snake Ladder Game
20 × 20
[0][9]
[8][0]
[r][c]
coordinate
A → B

↳ Requirements

↳ 1 Board (1, 100)

↳ Players → 2 players ✓

↳ Dice (1-6)  ✓

↳ Jumper ✓

↳ Game → logic of the game

↳ Coordinates ✓

↳ Main

↳ starting pt. of the game

1 Player class

5 mins

↳ name

↳ email ID

↳ contact no.

↳ address

↳ age.

→ set | get

② Dice

1, 2, 3, 4, 5, 6

MIN = 1

MAX = 6

2, 2, 3, 4, 5, 6

0.1

random — exclude

decimal ←

(0 - 1) * (MAX) + MIN

0 * ___ = 0 + 1 =

(int) Ans- range (0, 2)

0.5 * (6-1)

0.5 * 5 + 1 ⇒ 2.5 + 1 = 3.5

double d = Math.random() * (MAX) + MIN

return (int) d; ⇒ 3

0.7 * 5 + 1

3.5 + 1 = 4.5 ⇒ 4

0.12 * 5 + 1 ⇒ 1

Coordinate Class

↳ row, col

↳

Coordinate (int row, int col)

{

this.row = row

this.col = col;

}

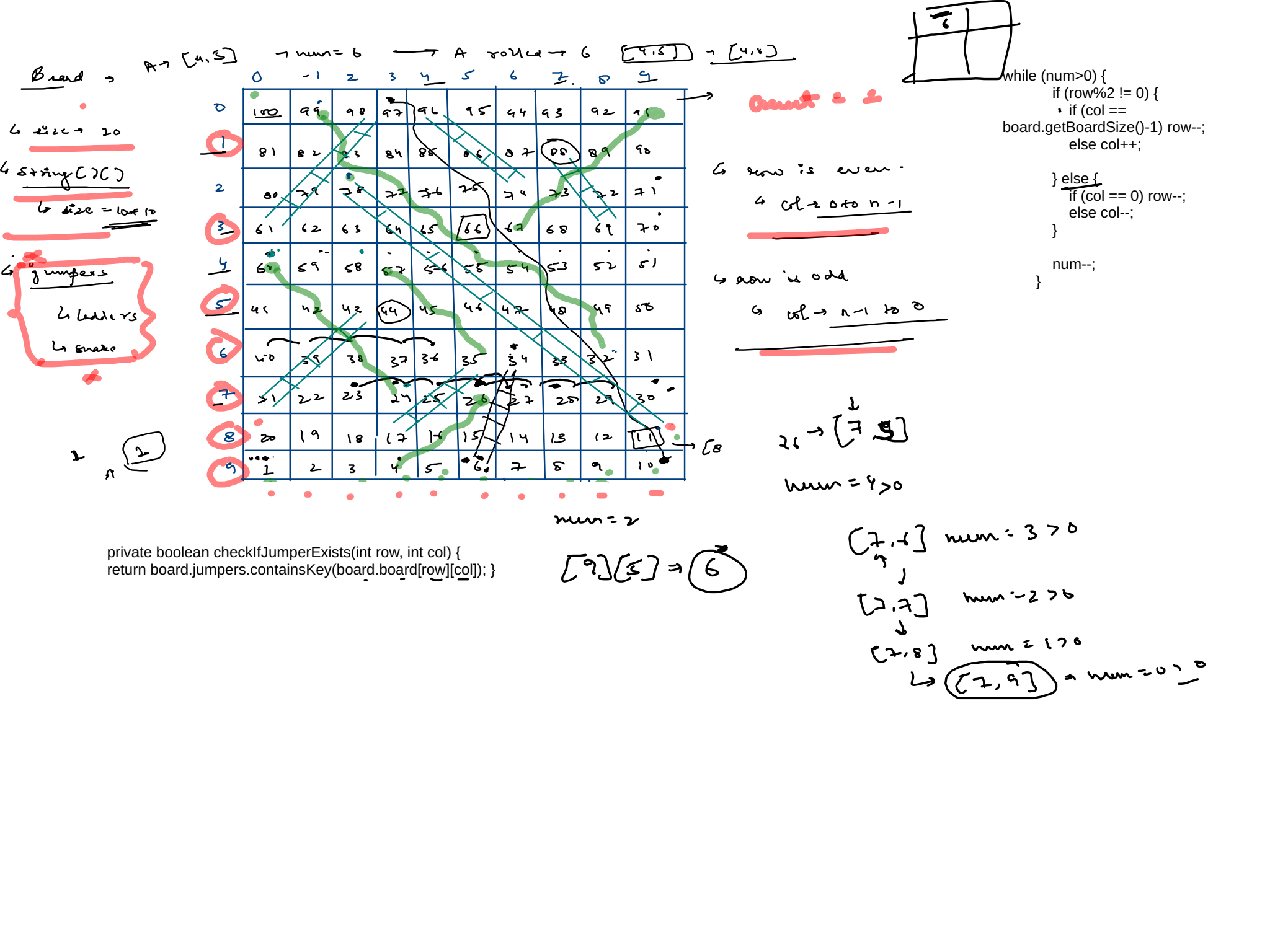
Jumper

- Coordinates → start → [0, 3]
- Coordinates → end → [7, 9]

start.row > end.row → ladder

- start.row → ladder
- start.row < end.row → snake

start.row < end.row → snake




```
while (num>0) {
    if (row%2 != 0) {
        if (col ==
board.getBoardSize()-1) row--;
        else col++;

    } else {
        if (col == 0) row--;
        else col--;
    }

    num--;
}
```

```
private boolean checkIfJumperExists(int row, int col) {
return board.jumpers.containsKey(board.board[row][col]); }
```

for jumper info

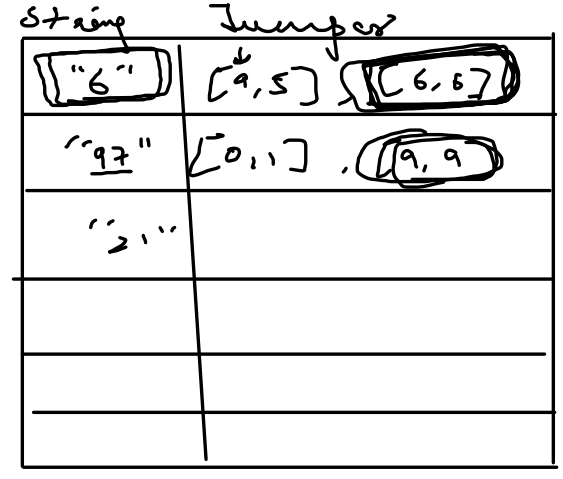
↳ Hashmap

↳ String

key

value

↳ jumper



9 > 6 → ladder

9 > 6 → ladder

0 > 9 → Snake

Hashmap < String , Jumper > map = new HashMap <> ();

Coordinates

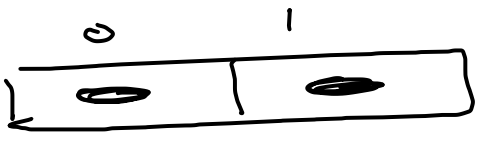
↳ start, end

9 > 6 → Ladder

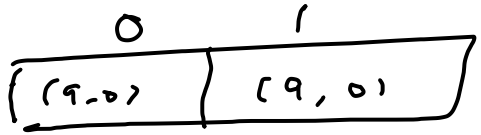
0 > 9 → Snake

Game

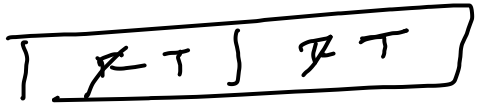
↳ Board ✓

↳ Players → 

A B

↳ Coordinates → Initial coordinates → 

↳ turn → 0, 1

↳ boolean isAllowed() → 

↳ A, B

F F

T

↳ dice →