

Snake Ladder Game



A → 1
 B → 6
 C → 4
 D → 4

	0	1	2	3	4	5	6	7	8	9
0	100	99	98	97	96	95	94	93	92	91
1	81	82	83	84	85	86	87	88	89	90
2	80	79	78	77	76	75	74	73	72	71
3	61	62	63	64	65	66	67	68	69	70
4	60	59	58	57	56	55	54	53	52	51
5	41	42	43	44	45	46	47	48	49	50
6	40	39	38	37	36	35	34	33	32	31
7	21	22	23	24	25	26	27	28	29	30
8	20	19	18	17	16	15	14	13	12	11
9	1	2	3	4	5	6	7	8	9	10

→ [0][9]

Snake

Ladder

Game

20 x 20

↳ A) B

[x][c]
_ _

↳ coordinate

→ [8][0]

99 98 97 96 95 94 93 92 91 90
 89 88 87 86 85 84 83 82 81 80
 79 78 77 76 75 74 73 72 71 70
 69 68 67 66 65 64 63 62 61 60
 59 58 57 56 55 54 53 52 51 50
 49 48 47 46 45 44 43 42 41 40
 39 38 37 36 35 34 33 32 31 30
 29 28 27 26 25 24 23 22 21 20
 19 18 17 16 15 14 13 12 11 10
 9 8 7 6 5 4 3 2 1 0

A
 B

↳ Requirements

↳ 1 Board (1, 100)

↳ Players → 2 players ✓

↳ Dice (1-6)  ✓

↳ Juniper ✓

~~main~~ ↳ Game → logic of the game

↳ Coordinates ✓

↳ Main

↳ starting pt. of the game

①

Player class

↳

name

↳

email ID

↳

contact no.

↳

address

↳

age.

5 mins

→

set | get

②

Dice

↳ 1, 2, 3, 4, 5, 6

MIN = 1
MAX = 6

2, 2, 3, 4, 5, 6



decimal

random
↳ (0 - 1) * (MAX) + MIN
on — = 0 + 1 =

(int) rand - 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

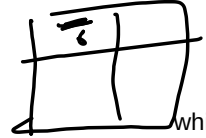
↳ Coordinates → start → [0, 3]
↳ Coordinates → end → [7, 9]

start.row > end.row
↳ ladder

[start.row > end.row → ladder
start.row < end.row → snake]

start.row < end.row
↳ snake

Board \rightarrow A \rightarrow [4,3] \rightarrow num = 6 \rightarrow A rolled \rightarrow 6 [4,5] \rightarrow [4,6]



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

size \rightarrow 20
 5 + 15 [7][7]
 size = 10 x 10

jumper
 Ladder
 Snake

2
 1

	0	1	2	3	4	5	6	7	8	9
0	100	99	98	97	96	95	94	93	92	91
1	81	82	83	84	85	86	87	88	89	90
2	80	79	78	77	76	75	74	73	72	71
3	61	62	63	64	65	66	67	68	69	70
4	60	59	58	57	56	55	54	53	52	51
5	41	42	43	44	45	46	47	48	49	50
6	40	39	38	37	36	35	34	33	32	31
7	21	22	23	24	25	26	27	28	29	30
8	20	19	18	17	16	15	14	13	12	11
9	1	2	3	4	5	6	7	8	9	10

Count = 1

row is even
 col \rightarrow 0 to n-1

row is odd
 col \rightarrow n-1 to 0

26 \rightarrow [7,9]

num = 4 > 0

num = 2

[9][5] \rightarrow 6

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

[7,6] num = 3 > 0

[7,7] num = 2 > 0

[7,8] num = 1 > 0

[7,9] num = 0 > 0

for jumper info

↳ HashMap

↳ String
key →

value
↳ jumper

String	Jumper
"6"	[9, 5] [6, 6]
"97"	[0, 1] [9, 9]
"21"	

9 > 5 → ladder

9 > 6 → ladder

0 > 9 → Snake

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

↓

↓

Coordinates

↳ start, end

9 > 6 → ladder

0 > 9 → Snake

Game

↳ Board ✓

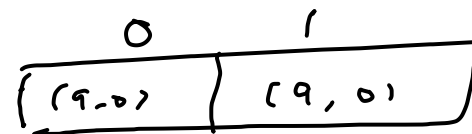
↳ Players →



✓
A

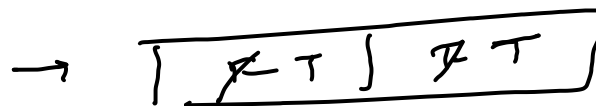
✓
B

↳ Coordinates → Initial Coordinates →



↳ turn → 0, 1

↳ boolean is allowed()



✓
A, B
F F
T

↳ dice →