Modified Snake Ladder Game

Problem statement

It's an online game and each game has multiple players and multiple games can exist at a time.

i. createGame

- Each board is a square matrix and given board size N, you will create N*N number of cells. Each cell is numbered from 1 to N*N (array of cells) and initially all players are in position 0.
- Snakes are represented here as Map. Key is a source cell number(from) and value is destination cell number(to). For example, it can be <10, 5> Means that snake mouth is at position 10 and snake tail is at position 5.
- Ladders are similar to snakes but the other way around. Here it can <40, 55>
- Return a unique game id. You can use Utils function to create unique id

ii. holdDice

- Behaves like pressing the buzzer in a quiz competition. Like whoever presses the buzzer first, they get a chance to answer the question first. Similarly, a buzzer is exposed on each of the player's UI, each player will simultaneously call holdDice method to acquire the dice. Allocation logic is simple that whoever reaches your system first, they get the dice.
- Return true only If allocation is succeeded

iii. rollDiceAndMove

- Check for valid dice holder
- Generate random number between 1-6 which is a dice value and move player on the board, check for snakes and ladders
- Any player reaches end position(N*N) exactly, then a player won and the game ends
- Any player tries to move ahead of end position (> N*N), they will retain the old position only and they lost the turn
- Only one player is allowed in a cell. But this logic can change in the future dynamically for each cell. Means that each cell can have different max capacity of players. Your code should be in an extensible way to satisfy this.
- Once moved, release the dice for the next turn.

Contract:

```
interface GameApplication {

    /*
    return unique game id
    */
    String createGame(int boardSize, Map<Integer, Integer> snakes, Map<Integer,
Integer> ladders, List<Integer> playerIds);

    /*
    return false
        - if player not part of this game
        - if the game is already ended
        - if the game id doesn't exist
        - if dice already allocated
    return true if hold dice is succeeded
    */
    Boolean holdDice(String gameId, int playerId);

    /*
    return false
        - if any player who doesn't hold the dice calls this
        - if dice is not allocated
        - if the game is already ended
        - if the game id doesn't exist
    otherwise roll dice and move and return true
    */
    Boolean rollDiceAndMove(String gameId, int playerId);
}
```

Important note:

- It is mandatory to code in Java for SDE 3 and SDE 4 candidate
- Interview timing split - 15(intro + problem discussion) + 90(coding) + 15(follow up/evaluation) mins
- Above time assumption is based on not using any external libraries to generate boilerplate code. If that is allowed, it can be completed even faster. So the interviewer should expect a better solution given the scope and environment for coding.