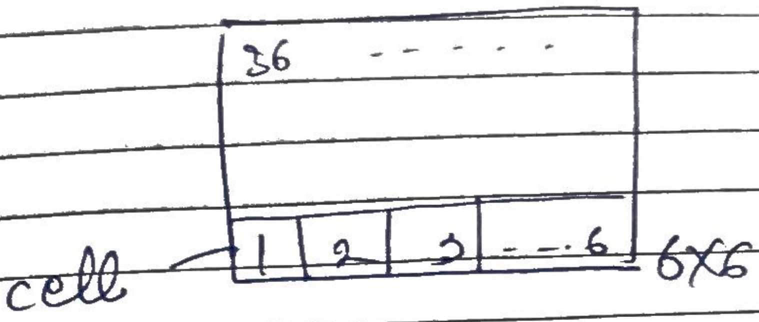
Snake n Ladder

Rough-flow



Requirement clarification

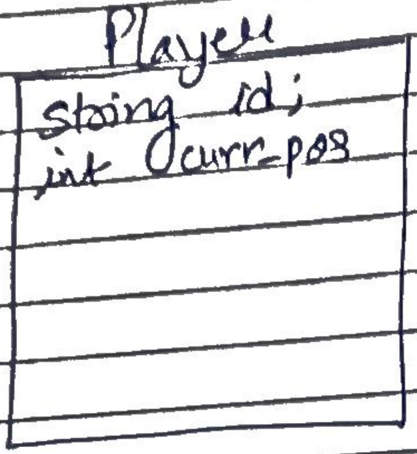
- How many dice?
  1, But could be scalable.
- How many snakes and ladder in game setup?
  Setup time → Snakes & ladder should be dynamically define
- What should be winning condition?
  ↳ Based on no of Players.

ent

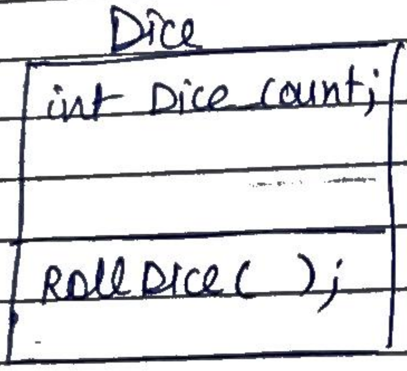
objects Identification

- → Dice
- → Snake, ladder
- → Board
- → Players
- → cells

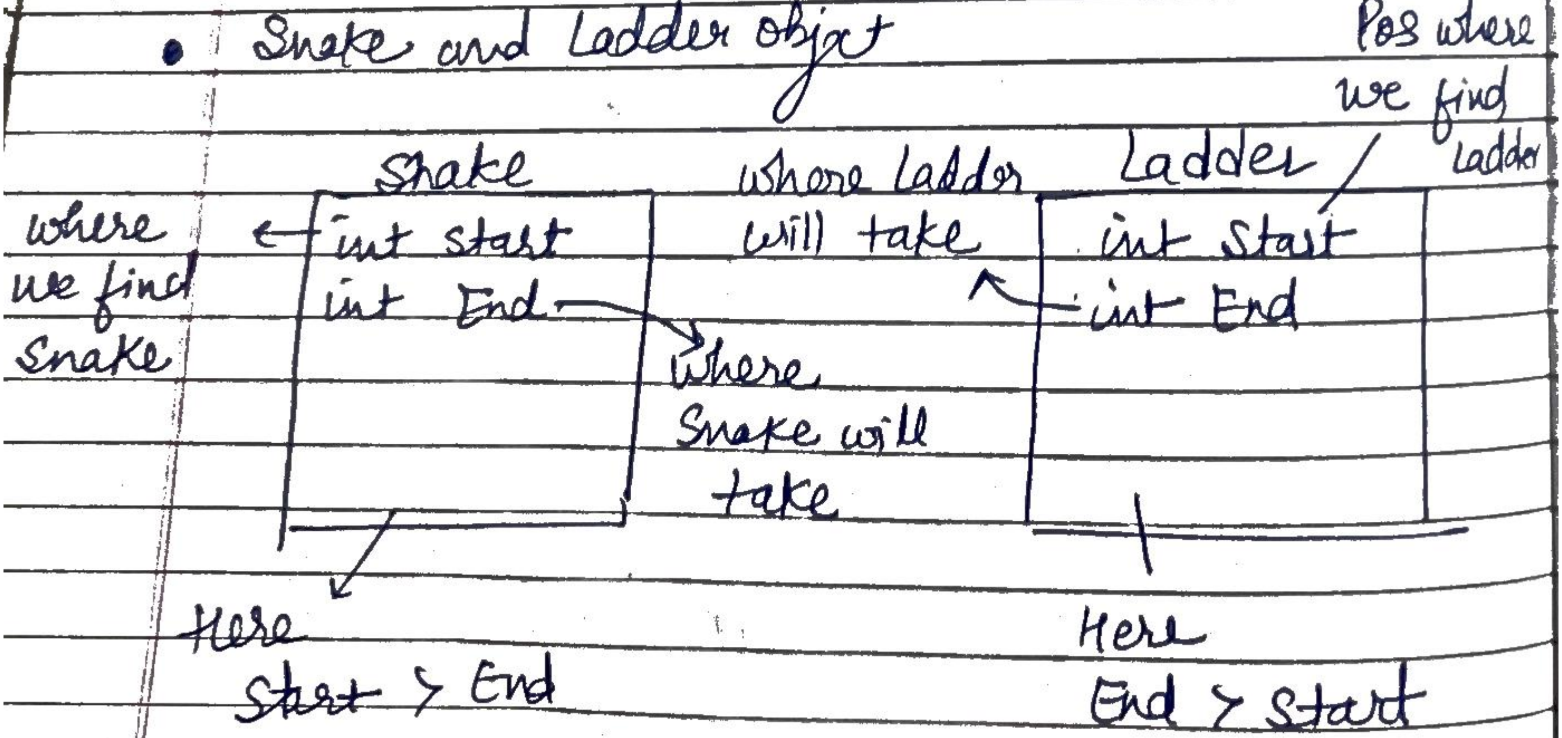
- Player object



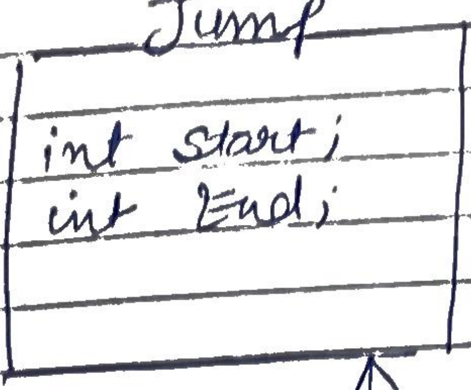
- Dice object



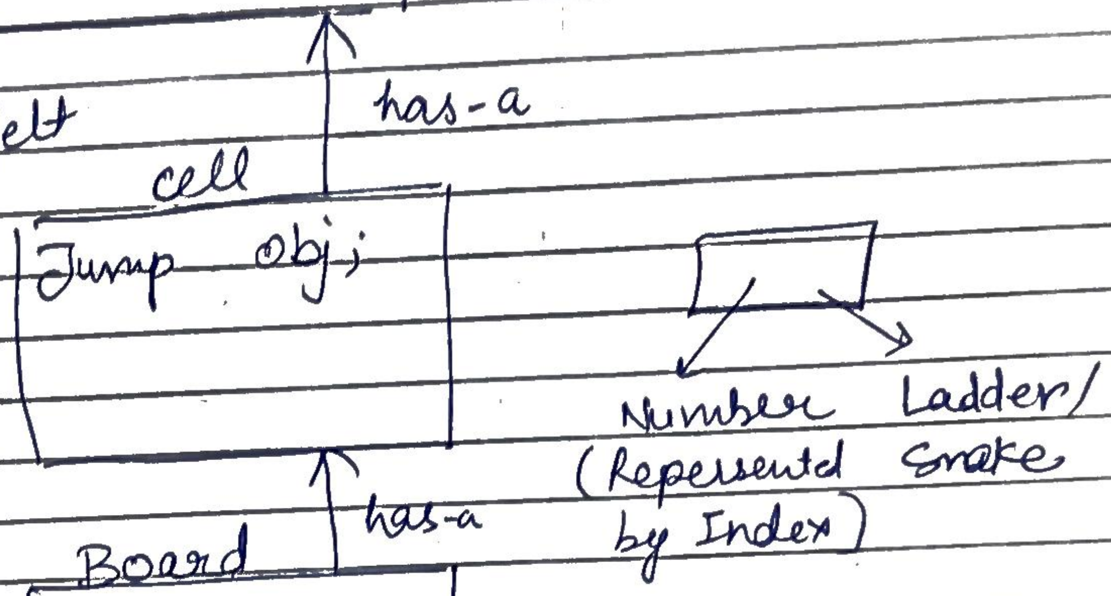
- Snake and Ladder object

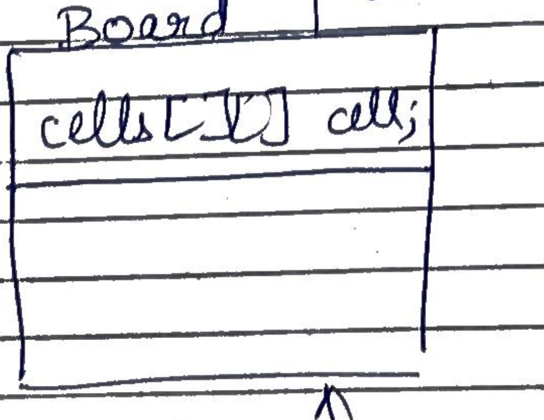


So instead of creating two objects we can create one object that combines both Snake and Ladder

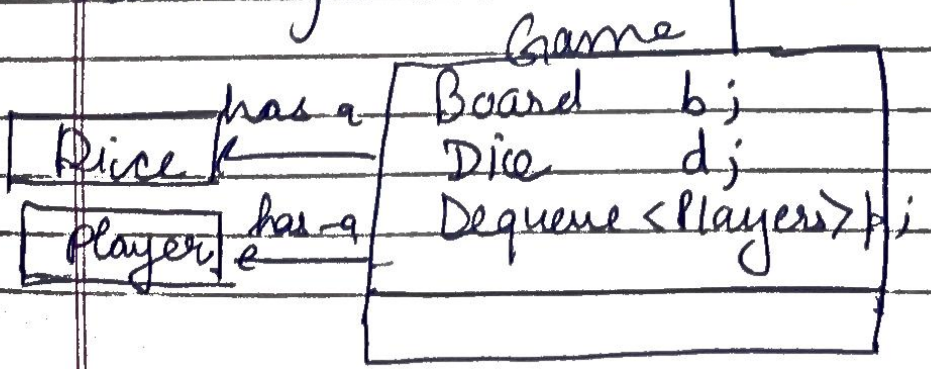


• Board object





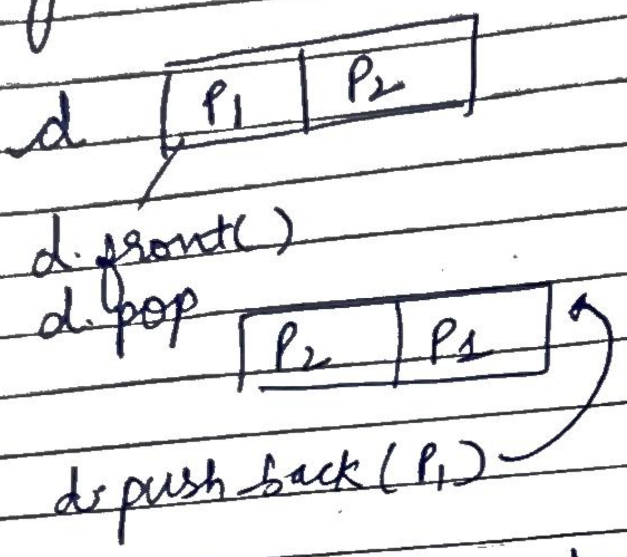
Game object



Page No.
Date / /

- Why Deque of playeres

Deque<Playeres> d;



It is easy to keep track of players and put them in correct order after each player has played it game.

- How to findPlayerTurn ?

It become easy by using dequeu to find player's turn

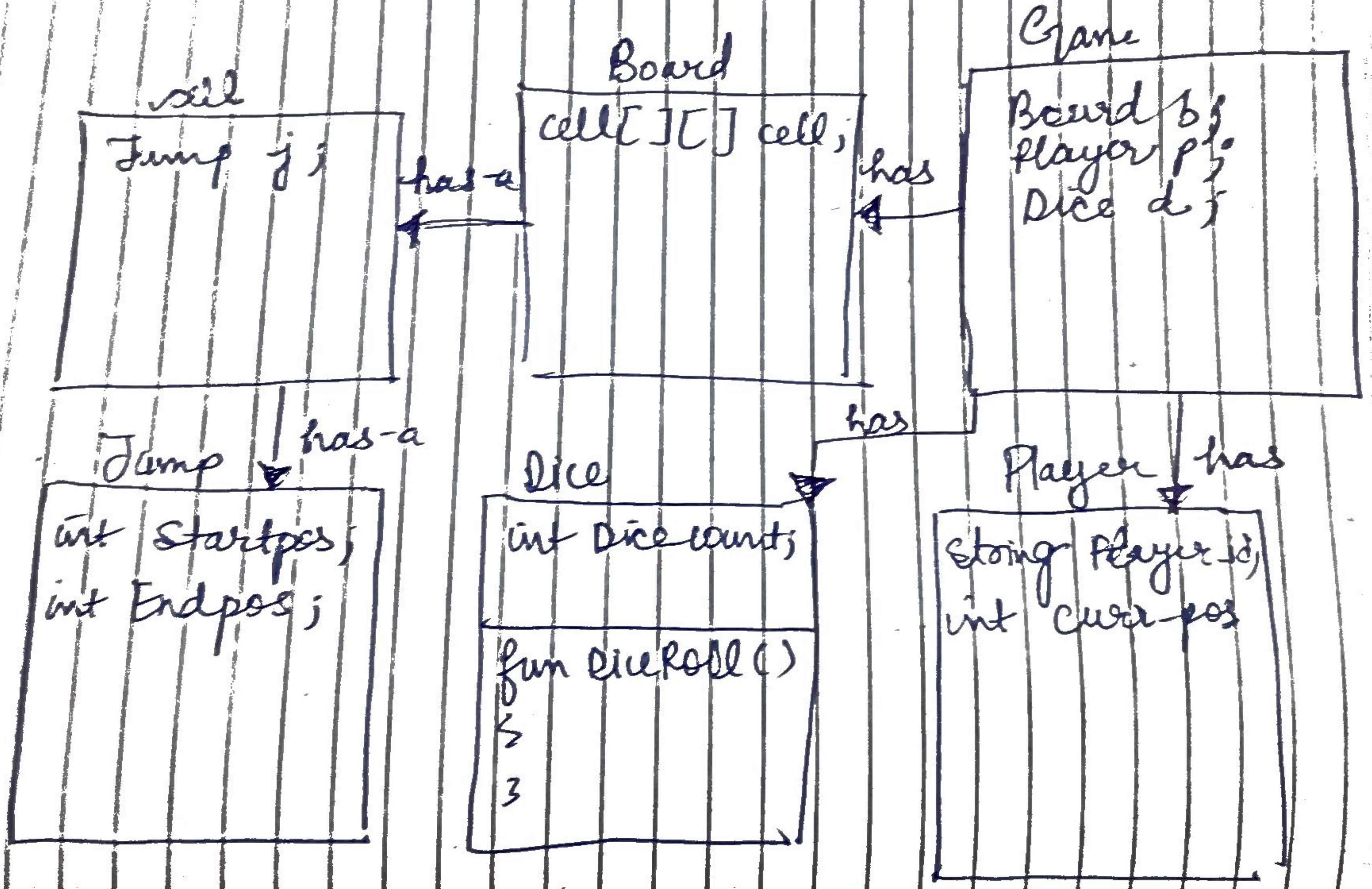
```
Player findPlayerturn() {
    Player x = playerlist.removefirst();
    playerlist.addlast(X);
    return x;
}
```

Then we will roll a dice and change the currpos + dice_number

Then we have to check whether we get jump either snake/ ladder ? we will jump from that if its present.

```
int JumpCheck( int newPos )
{
    if ( newPos > board.cell.length * board.cell.length-1)
      {
        return newPos ;
      }
 Cell  cell = board.getCell(newPos);
    if (cell.jump != Null && cell.jump.start == newPos)
      {
        return cell.jump.end; // if snake/
                                    ladder
      }
    else
        return Newpos;
      }
 }
```

High-level-View | UML



cell
Jump j;
has-a
Board
cell[ ][ ] cell;
has
Game
Board b;
Player p;
Dice d;
Jump
has-a
int Startpos;
int Endpos;
has
Dice
int Dice count;
fun DiceRoll()
{
}
Player
has
String Player id;
int Curr-pos