Snakes & Ladders Goal:

↳ be the first player to land precisely on square 100 by throwing two die

Some Rules:

1. When a player lands on a square with a ladder, the player gets transported up to the square at the end of the ladder

2. If a player lands on a square with a snake the player slides down to the square where the tail of the snake ends

3. If the player throws doubles, they may throw again

4. if you over shoot the 100th squdre you move back ~~to~~ the # of squares squares still left on the dice

* Note: there are 4 players. *

# What are the Objects?

- Players
- The Board
- Snakes
- Ladders
- Dice → never specified how many sides create constructor, then use 6 sided!

Two movement fields } Nevermind (inherited)

## What is important for each Object?

**Players**: * Their position on the board

**The Board**: * Keep track of player objects on the board
* Keep track of whose turn it is / How many spaces they move / handle snakes & ladders

**Snakes**: * move a player down

**Ladders**: * move a player up

**Dice**: * Roll a value

What data will each object need? (Brainstorm)

**Player:**
- Player id? (1)
- Current Position (2)
- ~~Method to increment~~
- ~~Method to decrement~~
- Constructor where position is automatically 0

- getter for (1) & (2)

Really just need a ~~change~~ position set

~~Board~~

**Snake:**
- Tail position (1)
- Head position (2)
- ~~Getters for (1) & (2)~~
- Getters for (1) & (2)
- Constructor where (1) & (2) are set

Same Concept

**Ladders:**
- ~~Top~~ ~~Bottom~~ Bottom of Ladder (1)
- Top of Ladder (2)
- Getters (1) & (2)
- Constructor where (1) & (2) are set

**Dice:**
- The number of sides (set in constructor)
- A method to return a random # based on number of sides.

**Board**
- List of 4 players
- Two die
- List of existing snakes
- List of existing ladders
- A method to return dice roll
- A method to check the users current position (& act accordingly via the rules!)

How the game will go:

1. Each player starts on 1
2. Starting with player at Index 1, roll & move spaces, check if a ladder bottom is hit or a snake head.
3. if they are hit, move spaces respective to the (L)/(S) that was hit
4. Increment the current player
5. Repeat steps 2-4 until:

   → A) the current player becomes 5, set it to 1 & go back to step ②

   → B) ~~A player~~ The current player reaches ~~spot~~ positi~~o~~n 100; end game

Springboot - Maven endpoints

/start → initalize game data

/advance → move forward in game

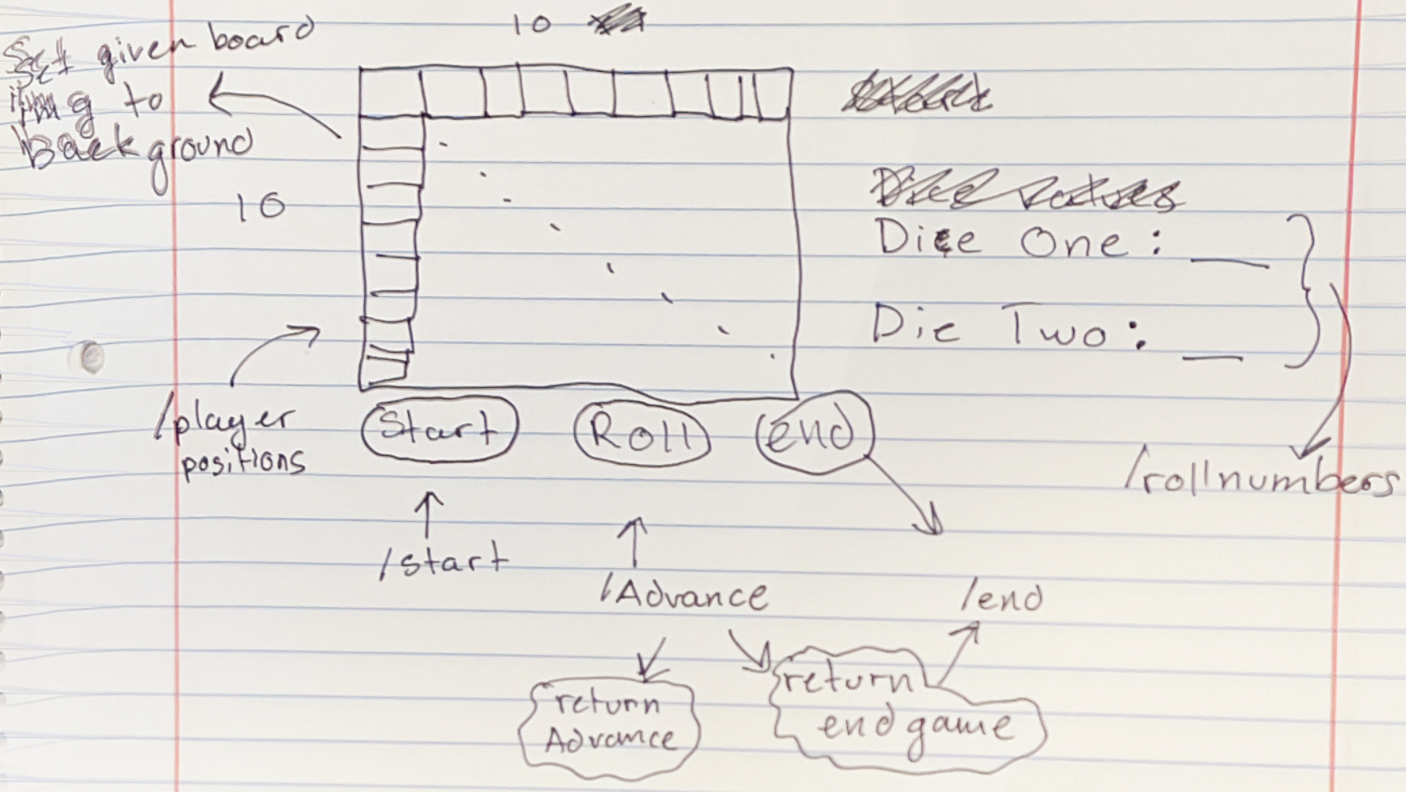
/error → throw an error

/playerpositions → returns player location on board

/rollnumbers → returns the value of the two dic on the board

/end → clear gamedata

For the front end:



h , t  top , bottom

| Which Snake Positions | | | Which Ladder Positions | | |
|---|---|---|---|---|---|
| * | 16 | 6 | * | 38 | 2 |
| * | 62 | 19 | * | 26 | 15 |
| * | 46 | 25 | * | 14 | 7 |
| * | 49 | 11 | * | 31 | 8 |
| * | 64 | 60 | * | 42 | 21 |
| * | 74 | 53 | * | 44 | 36 |
| * | 89 | 68 | * | 84 | 28 |
| * | 92 | 88 | * | 67 | 51 |
| * | 95 | 75 | * | 98 | 78 |
| * | 99 | 80 | * | 94 | 87 |
| | | | * | 91 | 71 |

In game board component, if I can keep track of player position

↳ I can use ngIf; ~~for example~~

Given: [1, 2, 3, 4]

playerposition1 = 1
playerposition2 = 2
3 = 3
4 = 4

} if the value of the variable = grid num set visible

---

What is there left to do?

1. Turn player 1 numbers into circles
2. make roll dice automatically refresh the other components
3. Update documentation

1 → Player Info, Game board ~~both~~ need to be updated upon rolling dice.

Finally call this refresh within dice roll

↳ Seperate service?

↳ Create behaviour subject as observable for:
- current player
- player ~~Posi~~ Positions
- Winner
- playerposition(1→4)
- Dice rolls

} Http get

↳ have a refresh button within here (use .next())

↳ ~~Inject~~ Inject the service in respective components

What needs to get done on the Frontend (order by priority:

1. Organize/ clean code ✓
2. Figure out why dice data will not immediately display? ✓ } on second touch it displays ???
3. Add player components! ✓ (didn't need a component)
4. Link the endgame endpoint! ✓

---

Working example exists!

* Don't forget to fix code logic such that the roll, on a double, does not skip forward *

For #2, may be the case that

↳ we retrieve rolls before game advances! ✓

↳ I have to look up HttpClient.get() examples & implement them! ✓

What would I like to ~~display~~ display for player-info?

- Current Player
- Player Positions
- If a winner, return winner!