Week 4 Discussion

Wednesday, 10/23/19

Reminders

PSA3 Checkpoint due Tuesday, October 29 11:59pm

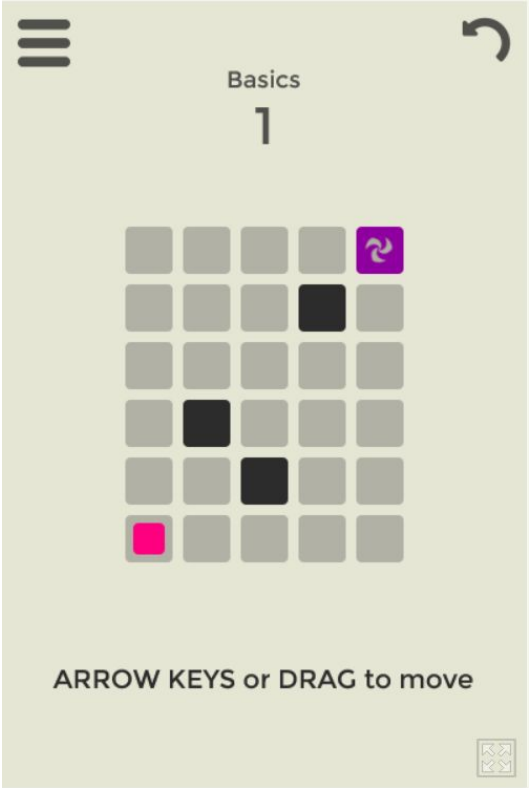
PSA3 Final Submission due Tuesday, November 5 11:59pm

Today’s agenda

- Introduction to Streamline
- GameState.java and its methods

What is Streamline

- Streamline is a puzzle game
- Want to navigate to the end
- Previously traveled tiles become obstacles



Demo

GameState.java

- Provided instance variables
  - `char[][] board`
  - `int playerRow`
  - `int playerCol`
  - `int goalRow`
  - `int goalCol`
  - `boolean levelPassed`
- **DO NOT ADD** any additional instance variables to this file.

Constructors

- Detailed Constructor: `public GameState (int height, int width, int playerRow, int playerCol, int goalRow, int goalCol)`
  - Initialize board with given parameters and other instance variables
- Copy Constructor: `public GameState (GameState other)`
  - Given another GameState and initialize instance variables based on that other GameState
  - Be sure to do a **deep copy** for arrays (a new array!) instead of just pointing to the same one

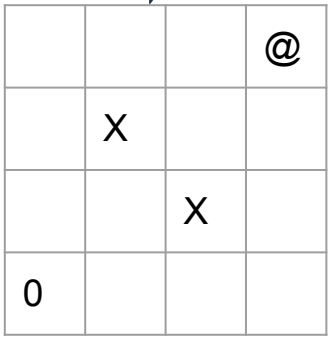
public GameState (GameState other)

other

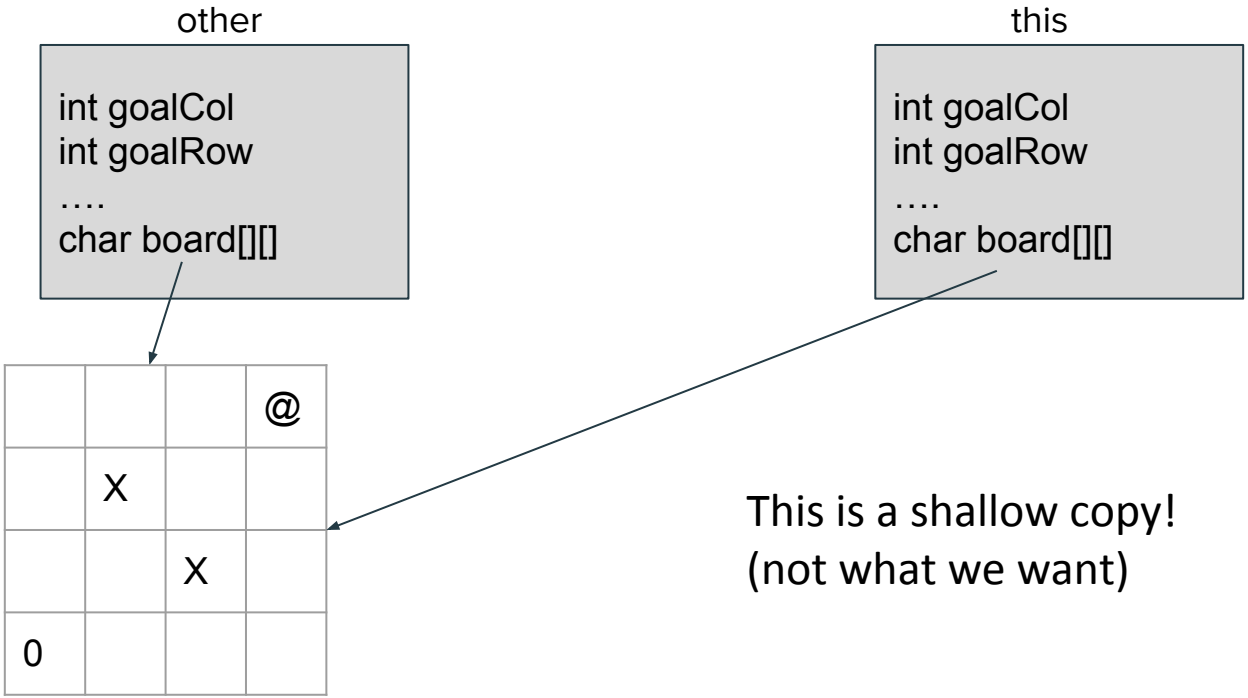
int goalCol
int goalRow
….
char board[][]

this

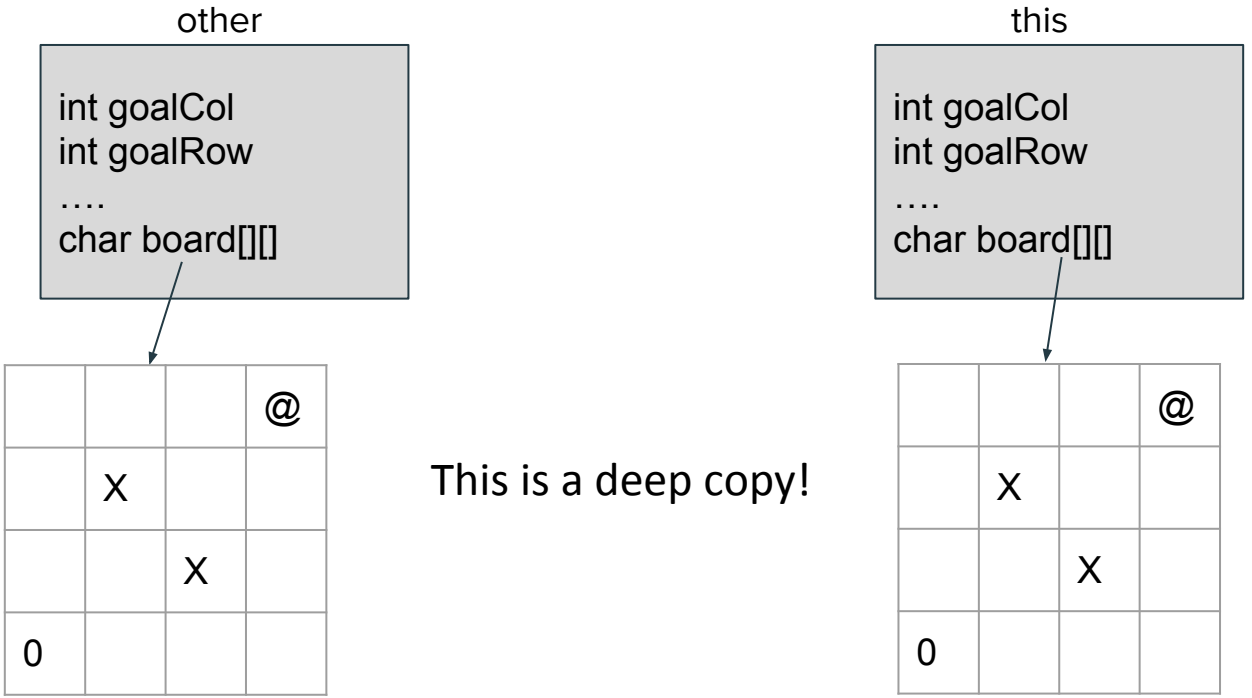
int goalCol
int goalRow
….
char board[][]



public GameState (GameState other)

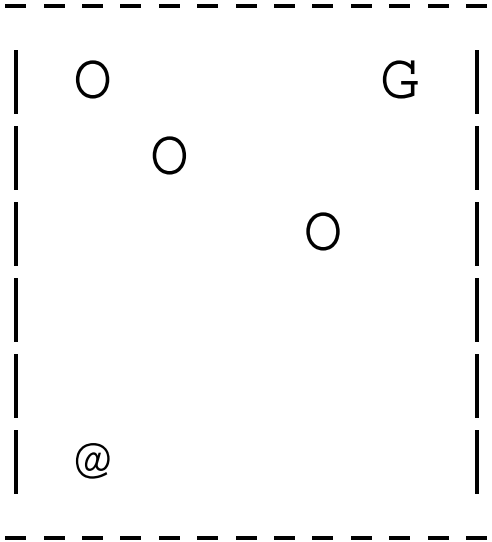


public GameState (GameState other)



public String toString()

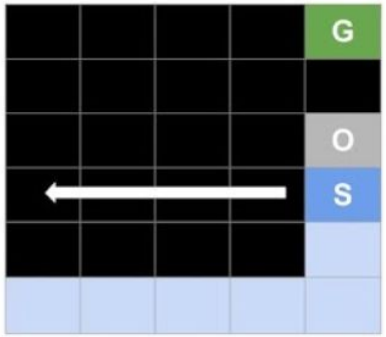
- Return a String representation of the board
- Implementation Idea:
  - Add the top border
  - Add row by row and column by column, making sure to add the goal and player when found
  - Add the bottom border



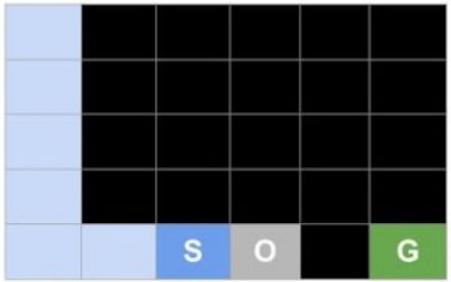
`public void rotateCounterClockwise()`

- Rotate the game board counterclockwise once
- Change the instance variables as needed based on rotation (positions of player, goal, etc.)

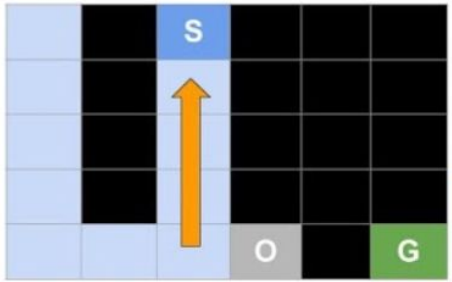
Why are we rotating our board??



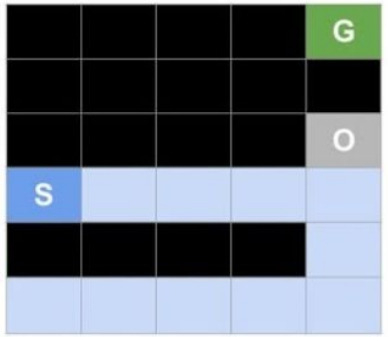
How do we move left?



Step 1:
Rotate board counter-clockwise 3 times



Step 2:
Move up



Step 3:
Rotate the board counter-clockwise

Review of 2D arrays in Java

<dataType> [ ] [ ] <arrayName> = new <dataType> [rows] [columns];

Ex. int [ ] [ ] table = new int[4] [2];

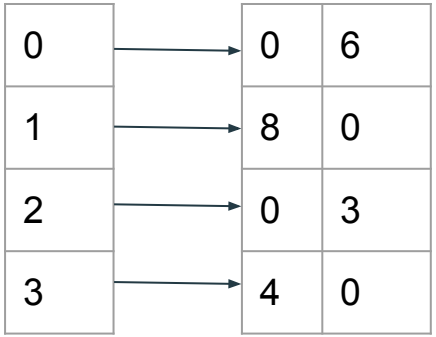
table[3][0] = 4; table[2][1] = 3;

table[1][0] = 8; table[0][1] = 6;

Index	**0**	**1**
0	0	6
1	8	0
2	0	3
3	4	0

Review of 2D arrays in Java

Memory diagram representing 2D arrays as multiple arrays >>>



Stores references if storing objects rather than primitive data types:

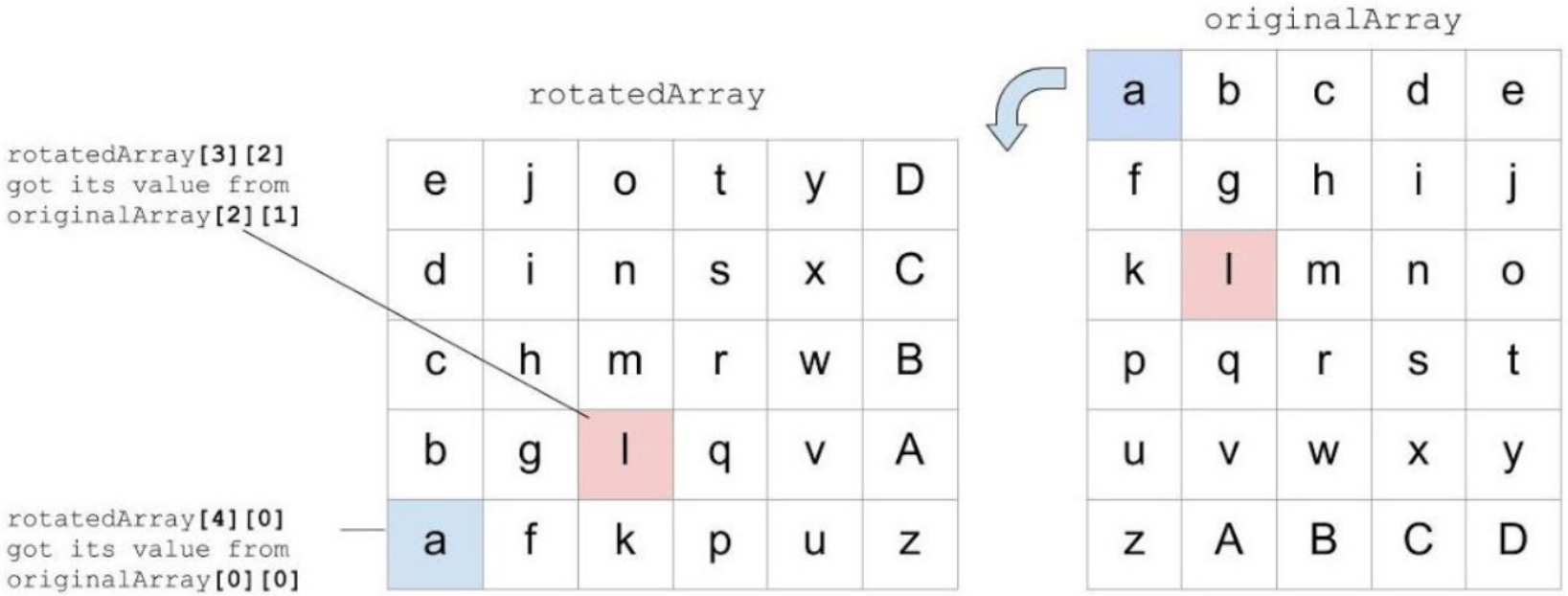
2D array

@A	@B
@C	@D

Heap

@A	{...}
@B	{...}
@C	{...}
@D	{...}

Rotating the Board



`public int countEmptyTiles()`

- Return the number of empty tiles within our game board
- How do we know if a tile is empty?
- This is a helper method that you can use for `addRandomObstacles()` and `addRandomZappers()`

`public void addRandomObstacles(int count)`

- Add `count` obstacles onto the board in random locations
  - How do we randomize the locations?
- Return immediately if `count` is less than 0 or there are `count` is greater than the number of available spots
  - Use `countEmptyTiles()`
- **Do not override player's position, goal position, or other existing entities.**
  - When placing obstacles on the board, only count the placement if it is an **empty** position.

`public void addRandomZappers(int count)`

- Add `count` zappers onto the board in random locations
- Similar to `addRandomObstacles()` but you must also randomize the zapper to add onto the board
  - How do we do this?
- **AGAIN: Do not override player's position, goal position, or other existing entities.**
  - When placing obstacles on the board, only count the placement if it is an **empty** position.

java.util.Random

- Allows you to generate pseudo-random numbers
- How can we test our code if it is generating random values?
  - Pass in your own seed
- Example:
  - `Random rand = new Random(123);`
  - `System.out.println(rand.nextInt(10)); // outputs 2`
  - `System.out.println(rand.nextInt(10)); // outputs 0`
  - `System.out.println(rand.nextInt(10)); // outputs 6`
- You are free to specify a seed in your code to test your methods.
- We will **not** be deducting points for using your own seed for your Random objects.

`public void moveLeft()`

- Move the player's current position left until it should stop
  - When should it stop?
- Leave a trail of `TRAIL_CHAR`s for all positions passed through
- Check to see if the player has passed the level
  - Set `levelPassed = true`, return
- Game ends when:
  - `playerRow == goalRow`
  - `playerCol == goalCol`
  - `levelPassed == true`
  - `board[playerRow][playerCol] == board[goalRow][goalCol]`
  - `board[playerRow][playerCol] == PLAYER_CHAR`

`public void move(Direction direction)`

- Use `rotateCounterClockwise()` and `moveLeft()`
- High level algorithm
  - Rotate some number of times to orient the Snake in the correct direction.
  - Move left.
  - Rotate back to the original board position/orientation.
- Look in `Direction.java` to help you determine how many times you have to rotate to get to the right orientation

`public boolean equals(Object other)`

- Compares calling object (`this`) and `other` object
- To override the `equals` method
  - If `other` is `null`, return false
  - If `other` is not of type `GameState`, return false
    - Use the `instanceof` operator
  - Check the contents of both objects
    - What should we check? Hint: Slide 6
    - All fields of both objects must be the same!