Hexagon Class

Brian Tsang

HW 04: Recursion Work Journal

- Hexagon Tiles
  - Contains a tile #
  - Color of the segments of the Hexa tiles
  - Position of

To get the info for the hexagon, we need to create getters and setters. For tile number, segments, and position.

First, we need our private variables
- private int numTile
- private int position
- private char[] segments

- Need to know the number of each tile of each hexagon (7)
- The position of where the Queen would move like a chess piece
- Setting the color in the segment of the hexagon

- Rotating Tiles
  - To rotate, we need to know the direction we are going.
  - Ex: We want to go North West, we get the color of that segment
  - Then move all segments one position to the right
  - Set the color of the specific direction it's facing.

We will be having 2 classes:
one for the hexagon itself
one for the board of the hexagon til

So we are almost done, for the hexagon class. Final method we need is a toStrings to organize how the output would look for the position and tile number.

Brian
Tang

Board Class

- Keeping track of the board
  - I would make a private Hexagon method that would get the tiles with a file name
  - Read and returns the data
- So overall this method reads how the tile layout is from the given text file.

- Tracking Tile Position
  - design a method with an hexagon object.
  - compare using the center hexagon while setting the segments
  - compare other hexagons with ~~each~~ each other.

✓ - How do we know we find the solution?
  - First we make a private hash set method with three variables in the parameter.
  - Two objects, one integer for position (board)
- Then it will check if there are any positions to fill, find if any tiles are not assigned, set a place to save the position of the hexagon, and if it's valid it'll proceed to place the next hexagon.

* - Finding all solutions
  - create a hashset to store solution
  - uses a for loop to check solution by placing each hex at a tile
  - returns the solution stored
  - calls out method in the main.

X - What if no solution can be found in the given set?
  - At first hexagon class position is default set as "-1".
  - It will print out "no solution" when "solutions" variable is Empty().
  - We know in the other find solution ~~find~~ method where it compares if two adj. segments are the same color.

Brian Tang

Backtracking

- For this class, we should backtrack the tiles and the position its in.
- Set tiles at 0, find all solution
- resets position back to -1 for all tiles.

- Use it when it's not adj., no solutions.
- no solution, we have to reset

- What we should do for backtracking
  - Reset back to the very beginning of the tiles and position.
  - Back to the center tile and original position
  - Rotating to see if there's any adj. matches to go back to the center.
  - After rotating, go back to original position, take out the hexagon, and set back position to -1.

Partial Solution ⟶ X Actual Solution

- maybe same color but different tile or position vise versa
- solution either has to be right or wrong or it will not go through
- If it's partial, it'll still count as not a solution so the hexagon will go back to the center and reset the position.