Battleship Design

Author: Katana Bierman

CSC 120

Board Class:

Represents a single board in a game of battleship.

Methods:

- __init__(self, size):

  Requirements:

  - constructor-builds an empty board
  - all boards are square
  - assert size is positive

  Steps:

  1. assert size > 0
  2. make a list of board data for each coordinate
     1. each element will look like ((x, y), ship, hit)
        1. the ship is initially None for all coordinates
        2. hit is a Boolean of whether the ship has been hit. All initialized to False
  3. make self._size variable

- add_ship(self, ship, position):

  Requirements:

  - adds ship to board at given position
  - assert ship and position are within valid indices
  - assert ship does not overlap other existing ships

  Steps:

  1. find what all of the ships possible coords would be
  2. assert ship is in valid indices
     1. loop through possible ship coords. If any of the x or y values are greater than the size, return the original board coordinates set

3. assert ship does not overlap existing ships
    a. loop through all current board coordinates
    b. if the current coordinates are in the possible coordinates, but there is already a ship there, return the original board coordinates set
4. add ship to board
    a. loop through the ship's coordinates
        i. find the index of (coords, None, False) in board coordinates
        ii. change board coordinates[index][1] to the ship

- print(self):

  Requirements:

  - print the current state of the board

  Steps:

  1. print out the top border
  2. set y_num equal to the size of the board
  3. loop through the range of the size to print each row
      a. print the y_num and the left border
      b. loop through the board coordinates (nested loop)
          i. if the y_num is equal to the y in the current coordinates then print the spaces.
              1. If there's no ship, print a period. If the ship is sunk, print an X. if the ship is hit, print *. Otherwise, print the first letter of the ship's name
              2. If the coordinates equal (size, y_num) then print two spaces and the right edge
          ii. subtract 1 from y_num
  4. print bottom border
  5. print x coordinates

- has_been_used(self, position):

  Requirements:

  - check if given coordinates have been hit

  Steps:

  1. loop through the boards coordinates to find the ship data at the given position
  2. return the ship data at the second index (whether or not it has been hit)

- attempt_move(self, position):

  Requirements:

  - handle a shot taken at a given location

  Steps:

1. assert the position is at a valid location in the grid
2. assert the location has not been shot at before
3. change the spaces data at index 2 to True
4. if the space has None as its ship, print miss message
5. else change the hit value in the ship object's to True and print the hit message with the ship name

Ship Class:

Represents a single ship in the game

Methods:

- __init__(self, name, shape):

  Requirements:

  - Build new ship object

  Steps:

  1. Make variables
     a. Public variable self.name set equal to name
     b. Public variable self.shape that has a list with each element is the data of the ship as ((x, y), hit) where hit is a Boolean of whether that space has been hit
- print(self):

  Requirements:

  - Print out the status of the ship

  Steps:

  1. Loop through self.shape
     a. If hit is True, print *, otherwise print the first letter of the ship
     b. Print 10 – the length of self.shape spaces, then the name of the ship
- is_sunk(self):

  Requirements:

  - Determines if the ship has been sunk

  Steps:

  1. loop through self.shape

2. If any have the hit variable as False, return False
3. If the loop passes with all spaces hit, return True

- rotate(self, amount):

  Requirements:

  - rotate the given ship 90-degrees clockwise

  Steps:

  1. create an empty list
  2. loop through the index in the range(amount)
     a. loop through self.shape
        i. find the old ship's (x, y)
        ii. append (y. -x) to the empty list
  3. return a new ship object