Sunil Jain and Vatsal Parikh

Java Programming

Period 2

May 11 2020

Java Design Cycle Plan

Project: Go

General design:

* Main:
  + class calls starting window and runs the game.
* StartingGame:
  + Holds jframe that allows you to allows you to set the size of the board.
  + Start button calls the game with the int variable from a text area
* Game:
  + holds the game frame and game interactions. And it also sets the display settings for the frame.
    - Key listener for quitting
    - Mouselistener for placing points
      * Calls getPoint from GamePanel to determine if the point is clicked on.
* GamePanel:
  + Extends JPanel
  + Holds every aspect of game logic.
    - Ex: update board method that implements recursion to check if the point is surrounded or not.
  + Draws everything using paintComponent method
  + Holds getters and setters.
  + Holds all the math for drawing the board relative to the game screen size.
* Point:
  + Holds where it is on the board (x,y)
  + Holds where it's drawn on the screen (cx, cy)
  + Holds the diameter
  + Has a state variable to determine if it's BLANK, BLACK, or WHITE
  + Holds the owner variable to determine points.

Data structure

* Five classes: main, game, startgame, gamepanel, point
* Arrays and ArrayLists
  + Board (2d array of points) which will keep track of the points on the board and the blank spaces
  + Stack (1d arraylist of points) keeps track of the points of the same color that are connected and potentially captured
  + Board (1d arraylist of board) keeps track of the previous board patterns to prevent repeating patterns also known as the ko rule
* Objects
  + Point
    - Each stone on the board is on a point
    - Points are on the intersections of the lines
    - Has states: Black, White, Blank
    - If it is black or white draw a circle of the corresponding color on the point
    - If it is blank don't draw a circle on the point
    - Has its x y position to locate it
    - And a diameter variable to use when determining if the point got clicked
* Main variables for storing data
  + The main variables will be keeping track of the states of points
  + The x y locations of points
  + And variables to adjust the dimensions of the board and the frame

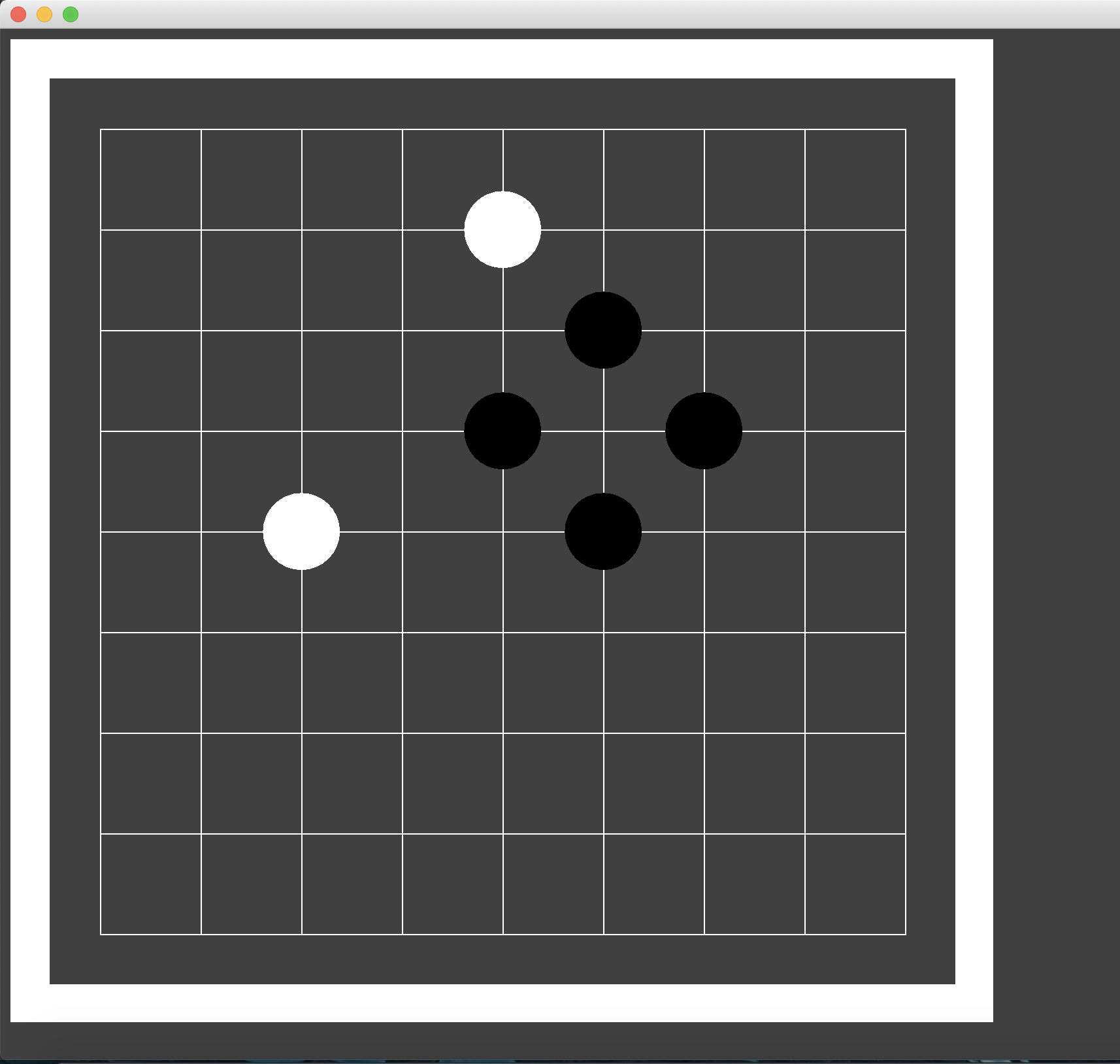
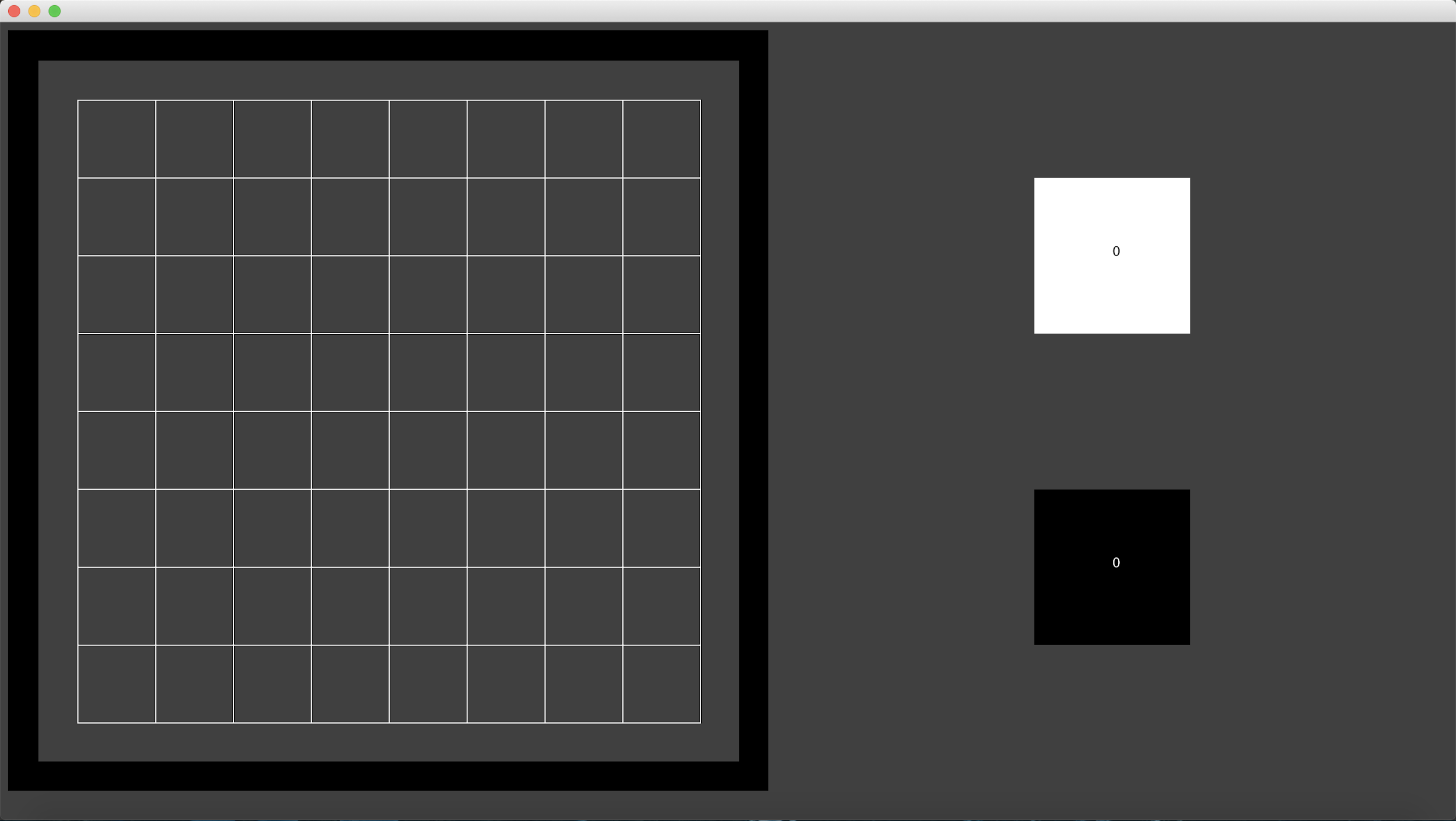
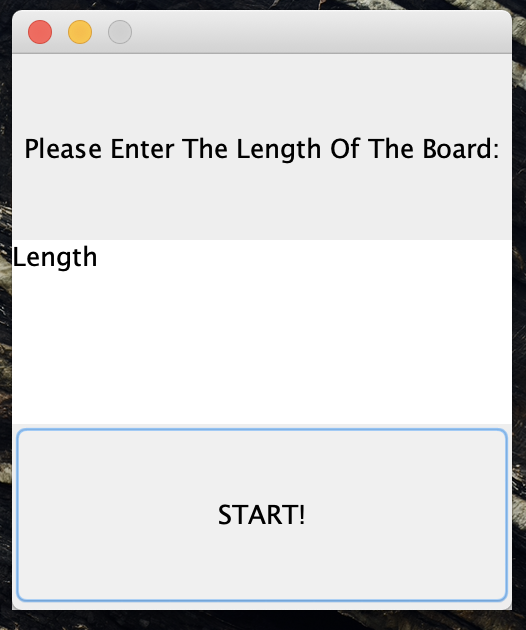
Algorithms:

* Main:
  + Main method
    - Calls and runs the start game and runs it.
* StartGame:
  + StartGame method
    - Basic setup of the StartGame window
      * Adds the label with instructions
      * Adds the text area for user input
      * Adds the button for running the game.
  + Mouse listeners
    - Checks to see if start button is clicked
      * If so then check if the textarea contains only numbers
        + If false then error message
        + If true then run the game class with the number from the textarea.
* Game:
  + Game method:
    - Basic setup of the frame
    - Add the game panel for it to draw the game and run the logic
    - Adds mouse and key listeners
  + Mouse pressed:
    - Calls getPoint from panel to get the point
    - If it's not null and the point’s state is blank
      * See if the move is valid
        + Yes:
        + Make the point whoever turns it is and switch the turn and update the board and repaint the frame.
        + No:

Error message

* + Key pressed:
    - If the player hits q then quit the game (for my convenience)
* GamePanel:
  + GamePanel method:
    - Updates variables
      * Int l for length
      * Calls setVars();
    - Make board with inputted length.
    - Fills board with new points
      * Sets the point’s x and y in the board to make it easy to track.
  + Validate play:
    - Checks to see if the passed in play is valid
      * Determines if the play makes the board look like any of the previous board states or gets into an infinite loop (uses boards variable).
      * For example, a black piece (a) is played and surrounds a point (b) and white has 3 pieces surrounding a and plays on b. This makes it so that point a is captured and disappears. Then black plays on point b and captures a and it's infinitely going to go on.
  + Get point:
    - Passes in ints x and y
    - Goes through board using double for loop
    - Check to see if passed in x and y minus a point’s x and y is within the point’s radius.
      * Yes:
        + Return that point
      * No:
        + keep on for looping until found
    - If not found after the double for loop then return null.
  + cleanStack:
    - Makes every point from board in stack blank and captured.
  + checkForCapture:
    - Passes in a board and a point and checks if the point is in the board then
      * Gets all it's liberties (up, right, left, down) of the passed point
      * If the liberty is the same state as the passed point and is not in the stack ArrayList then add that liberty to stack and call the checkForCapture method on that piece and keep repeating until the end of board is reached or end of the group is reached. Returns true if the group/point is captured if not captured it will return false.
  + updateScore:
    - Passes in what player to count for
    - Goes through the board
      * Counts for how many points are owned by the passed in player (captured by the passed in player)
      * Returns that number.
  + updateBoard:
  + setVars:
    - Sets variables:
      * Margin: divides the height of the screen by l and
      * Width: get as close to the height of the screen by incrementing l
      * Dl: d divided by l
  + paintComponent:
    - Draws a border that indicates whose turn it is
    - Draw white lines for the board.
    - Goes through the board
      * If the point is BLACK or WHITE then draw it on the board with the appropriate color.
  + Getters and setters.
* Point:
  + Getters and setters

Prototype



Better guide:

* <https://www.youtube.com/watch?v=xMshtO8h7RU>