* gameBoard
  + Creates window to display the game in
  + Draws in the game board by creating a multidimensional array with set dimensions of NxN
  + All of the cells begin empty
  + Has an icon for which player’s turn it currently is.
  + Has a location to display the score for each specific player which starts at 0
  + Has an int value remainingCells which displays the value set by findEmpty
  + Has a section which displays the current score for both players
* Player1
  + Creates player with a value for color which will display all of their pieces as that set color
  + Has set value for score which is saved to the player in between turns
  + Has set value for turn which allows playerMove to check if it is the player’s turn or not.
  + Has int value for invalidMove which saves whether or not the player has made an invalid move yet.
  + Has int value for the players score which is set by calcScore and displayed in the gameBoard window
* Player2
  + Creates a second player with a different value for color so that their pieces can be told apart
  + Has a value for turn which is set to 0 when their turn ends and sets the other players turn value to 1
  + Has a value for score which is set by the calcScore method and then displayed on the gameBoard as the game progresses. Default value is set to 0
  + Has int value for invalidMove which saves whether or not the player has made an invalid move yet.
  + Has int value for the players score which is set by calcScore and displayed in the gameBoard window
* playerMove
  + Runs an if statement to check see which player’s turn it is at the end of the loop.
  + Once the next player is determined it prompts the user to input their values for where the line will be drawn.
  + Runs ifValid in order to check if the line is valid or not. If the line is not valid it sets that player’s invalidMove value to 1 and ends the turn.
  + If the move is valid it then runs the isParallel method in order to check if the line is parallel to any other lines that are currently displayed on the gameboard.
  + If there are no parallel lines it then runs drawLine which will create the line on the gameboard of the specific player’s color. Any cells that this line passes through will also be changed to that player’s color which will be done by running swap for every cell on the line changing its color.
* ifValid
  + Checks to see if the values for all of the coordinates of the line that are inputed are within the area of the gameboard defined by NxN
  + Returns True if the play is valid
* drawLine
  + Creates line object with a loacation array of all the cells it occupies, a color value for what color it is, and a vector value for the vector of the line.
  + Runs through each point in the array along the line in a while statement.
  + Sets each individual cell along the line to the color of the player who is placing the line by running the method swap.
  + Reprints the gameboard with the new line added to the gameboard and passes the turn.
* isParallel
  + Checks the vectors of the input array with all of the current arrays on the board if any of them when divided by the input vector have a remainder of 0 then they are parallel and the method returns true.
* findEmpty
  + Runs through the entire array with a double if statement and checks to see which cells are empty and don’t have a player color. Initially set the value to 0 for each cell found it adds 1 to the value of remainingCells.
* swap
  + Swaps the current value of the cell to that of the player who’s turn it currently is.
  + Sets the cell’s color value to the player’s color value and reduces remainingCells by 1.
* calcScore
  + Runs through the array with a double if statement and adds 1 to each player’s score value for each cell with their color that it passes over.
* Play
  + Starts by running the gameBoard command which draws the window along with each of the individual components in the window.
  + Then creates the two player objects with the values of player color, player score, invalid moves, and player turn
  + Sets the value of the player color to a set default.
  + Sets the value of player score to 0 at the start.
  + Sets the value of invalid moves to 0 at the start.
  + Sets the value for playerTurn of player1 to 1 at the start and 0 for player2
  + Runs a while loop until either both players have an invalidMove int with the value of 1 or the remainingCells value is equal to 0.
  + Inside the while loop it checks to see which player’s turn it is and then prompts them for the start and end of their line.
  + From here it checks if the input is valid and if not it will set that player’s invalidMove int to 1 and pass the turn to the other player.
  + If the line is valid it will then check all of the current vectors of the lines on the gameboard with the vector of the input line and if any of the vectors on the gameBoard when divided by the input vector have a remainder of 0 then they are parallel.
  + If the line is parallel it will remove the line and pass the turn to the next player.
  + If there is no parallel line it will draw the line by checking each cell along the line swapping the color value of each cell to that of the player who’s turn it currently is if it is not already and adds 1 to the player’s score for every cell that has to be swapped.
  + After the line is drawn it runs the method findEmpty which will check to make sure that the board is not full before it prompts the next player for their turn. If remainingCells == 0 then it will run the method gameOver.
  + If remainingCells!= 0 it will pass the turn to the next player and run through their turn.
* gameOver
  + When either remainingCells == 0 || player1.invalidMove == 1 && player2.invalidMove == 1 then it will break the while loop and run the gameOver method.
  + The gameOver method will compare the player scores and display which player won the game or display tie if the scores are the same.