Lab 5: Drawing

Objective

- Draw geometric shapes in Java
- Use for loops
- Use conditional statements
- Use AWT and Swing libraries

What to hand in:

Submit your source code (.java files) to the dropbox no later than 11:59pm the night before your lab section's Lab 6. (Tuesday labs are due the following Monday, Wednesday labs are due the following Tuesday, and Thursday's labs are due the following Wednesday).

- Lab5.java
- ChessJPanel.java

Note: Late submissions will be penalized at a rate of 20% per weekday.

Marking Scheme:

Your solution will be assessed based on:

- Following lab specifications
- Commenting
    - Javadoc method comments
    - logic and variables commented
- Format and Conventions
    - appropriately named fields and methods
    - appropriate visibility of fields and methods
    - appropriate use of constants
- Compiles
- Matches given output
- Correctly handling user input

Steps:

1. Download the Lab5.zip file and un-compress it and open it in NetBeans.
2. **Complete the ChessPanel class**
    a. You are given a template for the Chess panel. Read through the comments in the file to understand what needs to be done.
    b. You should be able to compile and run the code as given
    c. First add the required class fields:
        i. constants for the number of rows and columns in a chess board (both 8).
        ii. constants for an x and y offset so the chessboard is not drawn on the top and left edge. An offset of around 30 should be good.
        iii. a variable to hold the size of the squares in the chessboard (which will be calculated based on the size of board the user selects).
        iv. All fields can be integers
        v. all class fields must be private
    d. Complete the constructor:
        i. The size of the grid squares is calculated based on the size of board passed in.

1. The grid squares size is calculated as the size divided by the number of rows or columns

e. Implement the `fillGridSquare` method
   i. This method will draw a single square given its row and column. It should call the `fillRect` function given the rectangle's x and y position, width and length. You'll need to calculate the absolute position of the square given its row and column as follows:
      1. xPosition = columnNum times squareSize plus xOffset
      2. xPosition = rowNum times squareSize plus yOffset
      3. the width and length are both equal to the squareSize
f. Test the `fillGridSquare` method by adding a call to it in the paint method
g. Implement the `drawRects` method
   i. This method will draw all the squares on the chess board by iterating through each row and column and checking if the parity of the row/column match. If they do, colour the square white, otherwise colour it black. Pseudocode is given in the `ChessPanel.java` file
h. Test your `drawRects` method
   i. add a call to it in the paint method
i. Complete the `paint` method, so that it:
   i. fill the whole panel with the colour black
      1. To set the colour to draw with, call the `setColor` method in the `Graphics` class. Remember you have an instance of the `Graphics` object called *g*. `setColor` takes a color as parameter. You can get a color from the Color class. `Color.BLACK` will return the colour black. `Color.WHITE` will return the colour white.
      2. you can get the width of your panel using the method `getWidth()`. You can call it by specifying: `this.getWidth()`.
      3. There is a similar method for getting the length of your panel
      4. To fill the panel, you can use the `fillRect` method from the `Graphics` class. It takes 4 parameters: x, y, width and height
   ii. call the function `drawRects(Graphics g)` to draw the chessboard
   iii. set the colour to White (see above)
   iv. draw a rectangle around the chessboard
      1. You can draw a rectangle using the `drawRect` method from the `Graphics` class. Like the `fillRect` method it takes 4 parameters: x, y, width and height

3. **Test your work using Lab5 Class**
   a. Run your program and try different size chess boards.
4. Once you are done (or while you are working ☺), **add JavaDoc method** comments to all methods you completed
   a. They should be of the format:

```
/**
 * Description of what the method does
 *
 * @param  param1Name  brief description as needed (add one line per parameter)
 * @param  param2Name  brief description as needed
 * @return      brief description (omit if void)
 */
```

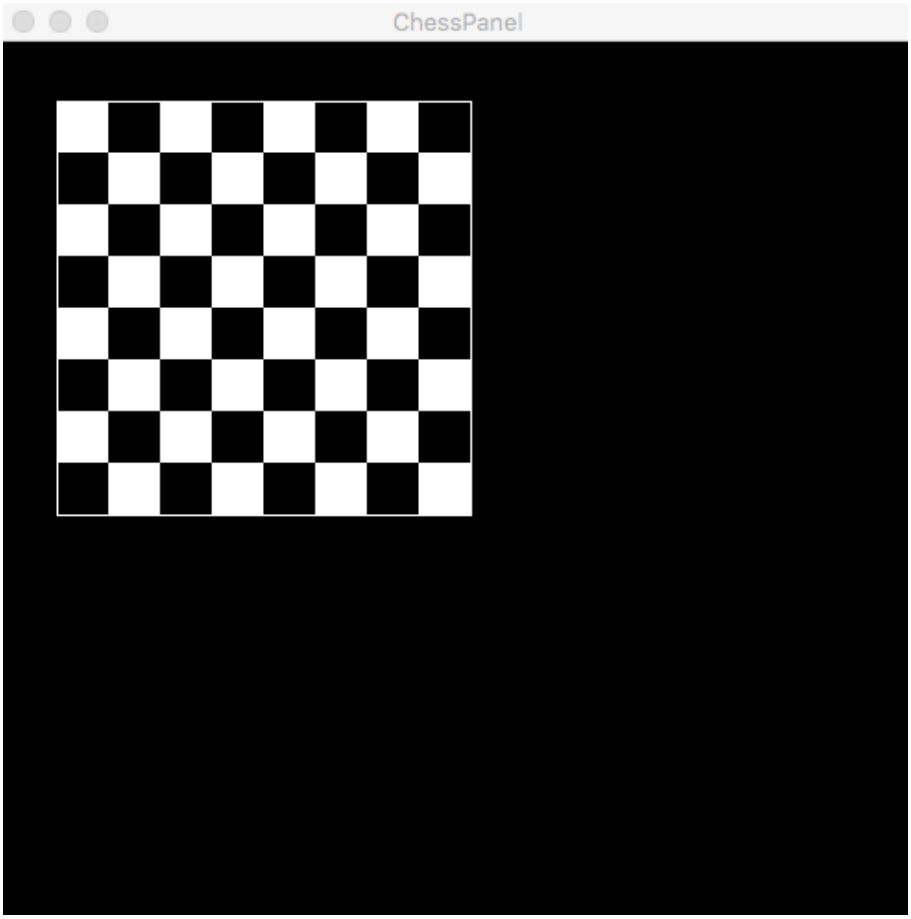
5. Add the date and your name as author to each file

NOTES:

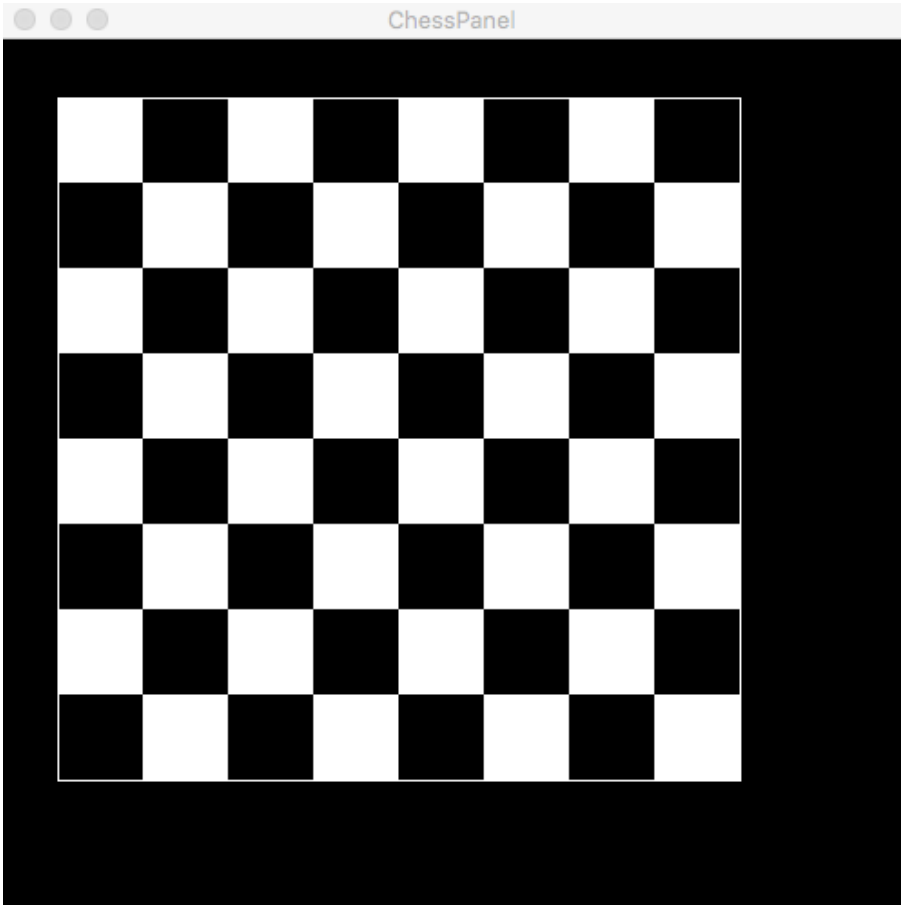
1. All class fields must be private
2. Do not change the signature of **any** methods
3. You may add any private methods you wish (but don't need to)

Sample output:

```
run:
Welcome to my chess-drawing program

How big a chess board do you want (less than 400) ?
-5
Sorry, that is not a valid size

How big a chess board do you want (less than 400) ?
401
Sorry, that is not a valid size

How big a chess board do you want (less than 400) ?
225
```




```
run:
Welcome to my chess-drawing program

How big a chess board do you want (less than 400) ?
376
```


ChessPanel