CSE 143X: Accelerated Computer Programming I/II

HW2: Café Wall (due Friday, October 9, 2015 11:30pm)

This assignment focuses method arguments and Graphics. Turn in the following files using the link on the course website:

- `Doodle.java` – A program that draws. . . something.
- `CafeWall.java` – A program that draws the Café Wall Optical Illusion.

This assignment is out of **20 points** (which will be scaled to be "worth a single assignment").

You will make heavy use of the `DrawingPanel.java` file. Make sure to download it and put it in the same directory as your `Doodle.java` and `CafeWall.java`. Also, make sure to include the following line of code at the beginning of your class file: `import java.awt.*;` We recommend you keep the documentation for `DrawingPanel` and `Graphics` open while attempting this assignment:

- https://courses.cs.washington.edu/courses/cse143x/15au/docs/DrawingPanel.html
- https://courses.cs.washington.edu/courses/cse143x/15au/docs/Graphics.html

Part 1: `Doodle.java` (2 points)

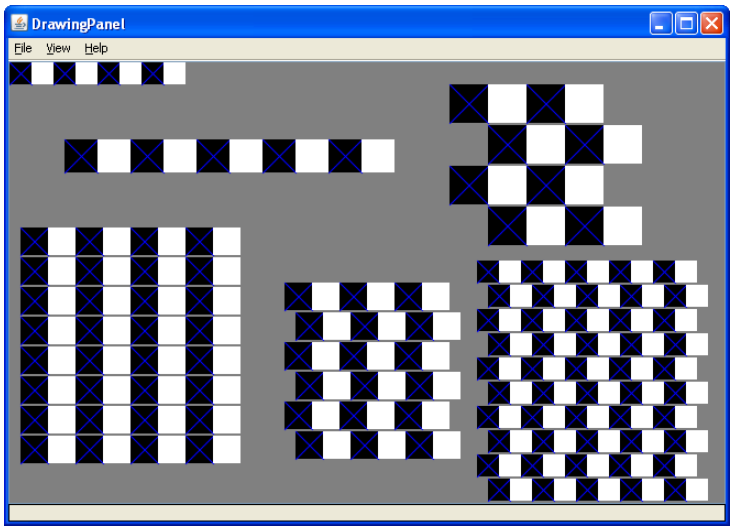
In the first part of this assignment, you will draw either a figure *or an animation* of your choice. You must follow these requirements:

- The figure must be at least 100×100 pixels
- The figure must contain at least three shapes
- The figure must use at least two distinct colors
- Your Part 1 and Part 2 may not be highly similar
- Your program may not have any infinite loops
- Your program may not read any user input

Your score on this part will be solely on external correctness. Additionally, *for this part only*, you may use whatever advanced material you like. The best images/animations will be displayed in lecture.

Part 2: `CafeWall.java` (18 points)

In the second part, you will produce the Café Wall Illusion which looks like this:



Café Wall has has a size of 650 pixels by 400 pixels and has a gray background. It has several levels of structure. Black and white squares are used to form rows and rows are combined to form grids. The output has two free-standing rows and four grids.

Producing A Single Row

Each row is composed of a certain number of black/white pairs of boxes where each black box has a blue X in it. The free standing rows in the output have the following properties:

Description	(x, y) Position	Number of Pairs	Size of Each Box
upper left	$(0, 0)$	4	20
mid left	$(50, 70$	5	30

We specify each row in terms of how many pairs of black/white boxes it has. Your program does not have to be able to produce a row that has a different number of black versus white boxes. The boxes should be specified using a single size parameter, because each should be a square.

You should have a single method that can produce any of these rows by varying the position, the number of black/white pairs, and the box size. You will need to use one or more for loops to write this code so that it can produce any of the various rows.

Producing A Grid

Once you have completed this method, write a method that produces a grid of these rows by calling your row method appropriately (you will again use one or more for loops to solve this task). Grids are composed of a series of pairs of rows where the second row is offset a certain distance in the x direction relative to the first. The output has four grids with the following properties:

Description	(x, y) Position	Number of Pairs	Size of Each Box	Second Row Offset
lower left	$(10, 150)$	4	25	0
lower middle	$(250, 200)$	3	25	10
lower right	$(425, 180)$	5	20	10
upper right	$(400, 20)$	2	35	35

Each grid is a square, which is why a single value (number of pairs) indicates the number of rows and columns. For example, as the table above indicates, the lower left grid is composed of 4 pairs. This means each row has four pairs of black/white boxes (8 boxes in all) and the grid itself is composed of 4 pairs of rows (8 rows total). Again, a single box size is used because each box should be a square. The offset indicates how far the second row should be shifted to the right in each pair. The grid in the lower left has no offset at all. The grid in the upper-right is offset by the size of one of the boxes, which is why it has a checkerboard appearance. The other two grids have an offset that is in between, which is why they produce the optical illusion (the rows appear not to be straight even though they are).

Brick Mortar

Each pair of lines in the grid should be separated by a certain distance, revealing the gray background underneath. This is referred to as the "mortar" that would appear between layers of brick if you were to build a wall with this pattern. The mortar is essential to the illusion. You should introduce a constant for the separation for the mortar, and you should set it to 2.

Restrictions & Testing

You are required to have the two methods described above (one for a single row, one for a grid). We will expect you to use good style (indentation, commenting, good variable names, etc) and to include a comment for the class and for each method.

You can use the DrawingPanel's image comparison feature (File, Compare to URL...) to check your output. Different operating systems draw shapes in slightly different ways, so it is normal to have some pixels different between your output and the expected output. If there is no visible difference to the naked eye, your output is considered correct.

For this assignment you are limited to the language features in Chapters 1 through 3G of the textbook. In particular, you should not use if/else statements.