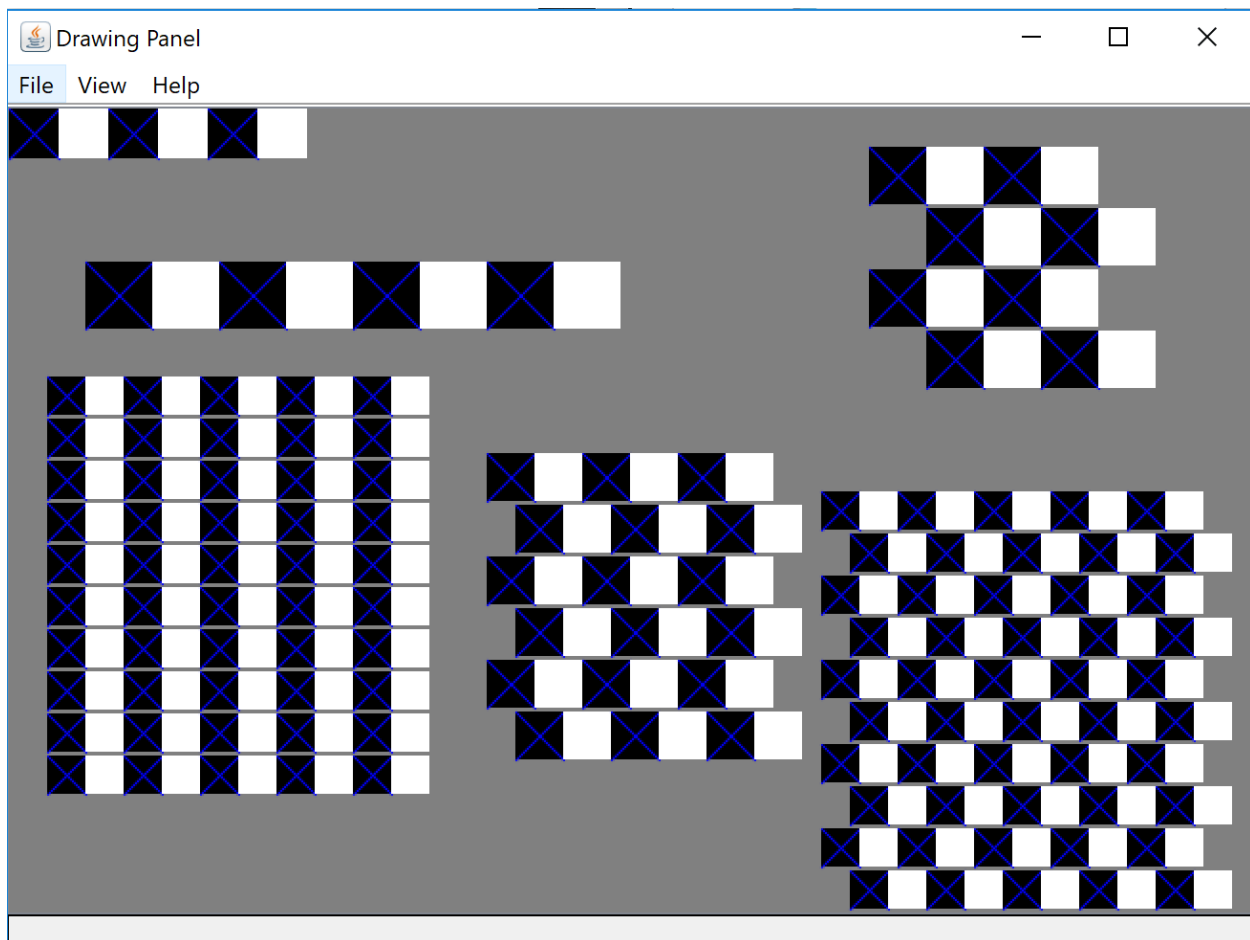


**EGR 222 — Software Engineering**  
**Homework #2 CafeWall**  
**Check Point Due: Saturday, 9/12/20, 11 PM**  
**Final Due: Wednesday, 9/16/20, 11 PM**

**[Section 1: Problem Specification]**

This assignment will give you practice with for loops, value parameters, and graphics. You will be using a special class called DrawingPanel written by Stuart Reges and his coauthor Marty Stepp. 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.**

In this assignment we will be exploring something that is known as the Café Wall illusion (as described in [http://en.wikipedia.org/wiki/Caf%C3%A9\\_wall\\_illusion](http://en.wikipedia.org/wiki/Caf%C3%A9_wall_illusion)). You are to produce the image below. It has a size of 650 pixels by 420 pixels and has a gray background.



This image 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. You should first write a method to produce 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:

<b>Description</b>	<b>(x,y) position</b>	<b>Number of pairs</b>	<b>Size of each box</b>
Upper-left	(0, 0)	3	25
Mid-left	(40, 80)	4	35

Notice that we specify each row in terms of how many pairs of black/white boxes it has. Your method doesn't have to be able to produce a row that has a different number of black versus white boxes. The boxes are specified using a single size parameter because each should be a square. You should develop 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.

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:

<b>Description</b>	<b>(x,y) position</b>	<b>Number of pairs</b>	<b>Size of each box</b>	<b>2<sup>nd</sup> row offset</b>
Lower left	(20, 140)	5	20	0
Lower middle	(250 , 180)	3	25	15
Lower right	(425, 200)	5	20	15
Upper right	(450, 20)	2	30	30

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. That 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). A single box size is again 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 figure 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).

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. Your

program should use 2 pixels of separation for the mortar, but you should introduce a program constant that would make it easy to change this value to something else.

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.

## [Section 2: Set up Your Project with IntelliJ]

0. Make sure you are using JDK8. In IntelliJ go to File → Project Structure and make sure Project SDK is set to 1.8 and language to 8. (See instructions on BB → Homework → Update to JDK8).

1. Click below and accept the invitation if you haven't.

<https://classroom.github.com/a/C9gG8IGw>

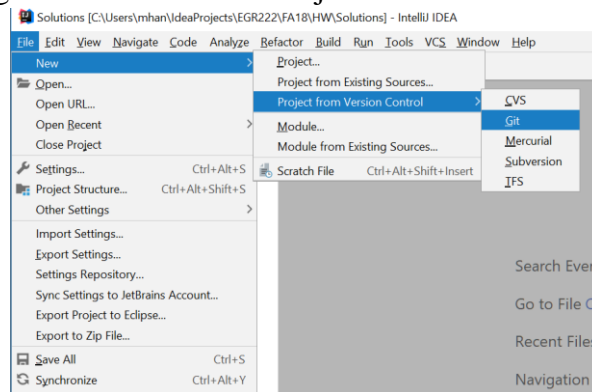
2. Go to your repository where the URL is

<https://github.com/cbu-egr222-fa20/hw2-YourGithubID>

Check you have contents (.idea/ png/ and src/ and other files).

3. Open IntelliJ (Do A or B depending on what you see there)

A) If you see below UI, go to File → New → Project from Version Control → Git



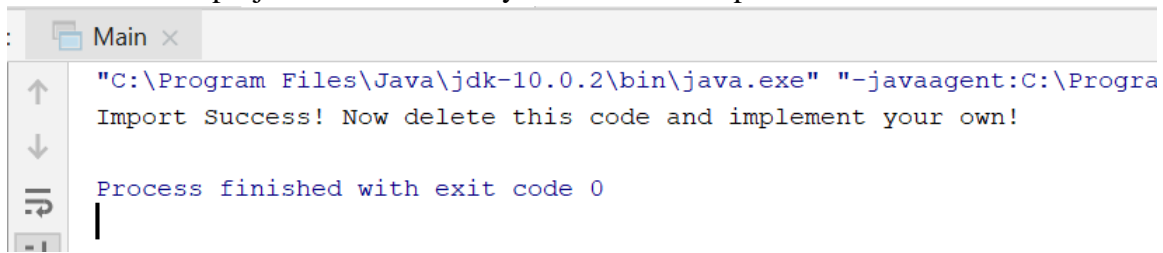
B) If you see below window, select “Check out from Version Control → Git”



3. Put the URL for your repository and hit Clone. Again, the URL is

<https://github.com/cbu-egr222-fa20/hw2-YourGithubID>

4. In the “Project” pane make sure you see all the contents you saw in 2.
5. Run main of this project and make sure you see below output.



```

: Main x
"C:\Program Files\Java\jdk-10.0.2\bin\java.exe" "-javaagent:C:\Progra
Import Success! Now delete this code and implement your own!

Process finished with exit code 0

```

6. You are ready to go! Implement your code in CafeWall.java. Make sure you update comments on top of CafeWall.java and also include comments as you implement your code.

### [Section 3: Submit your Code with Git Command Line]

**By Check Point Deadline, push your code progress to Github.**

**By Final Deadline, push your final code to Github and submit a Word doc with screenshots to BB.**

When you are done implementing code, submit to Github. The latest committed version before the deadline will be used for grading. Before you submit, please use the DrawingPanel's image comparison feature to check your output. The two expected output files are located under png folder (The mortar with 1 pixel is expected\_output\_mortar\_1.png and 2 pixels is expected\_output\_mortar.png). You must match the expected output within 500 pixels to be considered correct for both files.

1. Go to your terminal in IntelliJ and type below commands.

```
> git add --all
> git commit -m "Put your comments here"
> git push
```

If you need to make some modification and resubmit, you can simply repeat above 3 commands again.

2. Go to your remote repository where the URL is

<https://github.com/cbu-egr222-fa20/hw2-YourGithubID>

And make sure you can view your latest changes. Again, the latest committed version before the deadline will be used for grading.