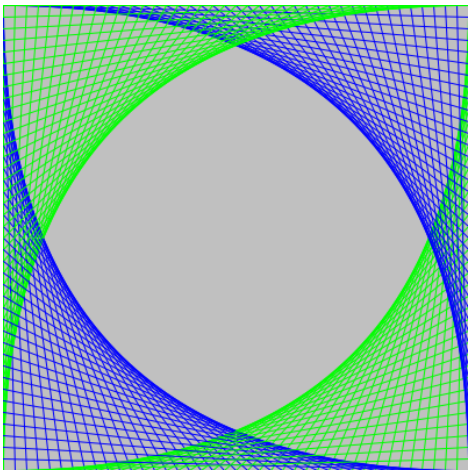
Project 2: Draw Curves from Straight Lines

Carefully read this *entire document* before beginning your work.

1 Objective

In this program you will use existing objects. You will also learn about and use graphics coordinates.

2 Input



This program gathers no input. No terminal/console output should be shown, either; only the graphics should show.

3 Output

3.1 Part 1

- Generate this graphic output shown at right, on a 400x400 DrawingPanel.
- Alter the colors to suit your taste (but use at least two different colors of lines and non-white background).
- Set the increment between lines to something pleasing to the eye; it does not need to exactly match the sample. Create an integer constant called LINE_INCREMENT for this purpose. I should be able to change that to a different number (evenly divisible or not) and see the expected output.
- For descriptions of the technique and samples of output, see these sites: site1, site2.

3.2 Part 2

- Create an additional drawing panel at least 600 x 600. On it, draw filled-in shapes of at least *two* types (rectangles, circles, etc.) in at least two colors.
- Each shape type should be drawn by its own function (that you write) with parameters indicating where it is to be drawn and how big it is to be. Include a color as a parameter as well; this will make the function more flexible. Create additional helper functions as you see fit.
- Draw several shapes (at least ten) in an attractive pattern; do something unique and interesting (e.g., draw a landscape or an eye-catching repeating geometric design).

4 Calculations

- Use the DrawingPanel.java class provided. Put this in the same folder as your solution.
- Use procedural decomposition to break down the program into logical pieces.
- Part 1: create and use class-level constants for the drawing panel size and the line increment.
- Part 1: try various line increments until you find one that is visually pleasing and looks curved. Note that the vertical increment will always be double the horizontal increment.

5 Code Implementation

Create a class called **GraphicsProject**; use this single class to do all your work. You should have only one main function that controls *both parts* of the project. Follow the Course Style Guide.

5.1 What You May Use

- Constants and variables; use the requested naming convention and declare them at the appropriate scope (i.e., use no global variables unless necessary, but global constants where they make sense).
- Definite loops (note that only *one* loop should be needed in Part 1).
- Drawing panel and graphics objects.
- Math methods like Math.round.

5.2 What You May Not Use

- Selection control structures, unless you are doing something "above and beyond" the basic spec. That means no if statements, switch statements, or ternary operators.

6 Submitting Your Work

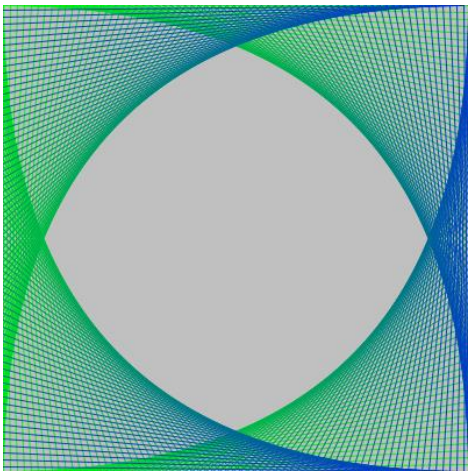
Submit your .java file; there is no need to submit the DrawingPanel.java code.

7 Hints

- Look over the supplied sample code that uses the DrawingPanel.
- Having a link to the online Java API reference will come in handy now and in the future.
- Do not duplicate code if it is avoidable; this is a good rule now and always.

8 Extra Credit: Gradients

For Part 1, instead of the drawing shown there, produce the one shown at right. Lines within quadrants are each a slightly different color. In this example, lines start with green but move slowly toward blue.



Your code must work with *any* two colors specified. Create Color constants START_COLOR and END_COLOR; make sure that any two colors you set work correctly. I will check your work by setting these to colors of my choosing.

9 Grading Matrix and Achievement Levels

Area	Percent
Part 1	
Drawing accuracy	15%
Coloring	10%
Looping	20%
Minimization of global variables	10%
General coding	10%
Part 2	
Shape and color requirements	10%
Function usage	15%
Documentation and style	10%
Extra Credit	3%
Total	**103%**