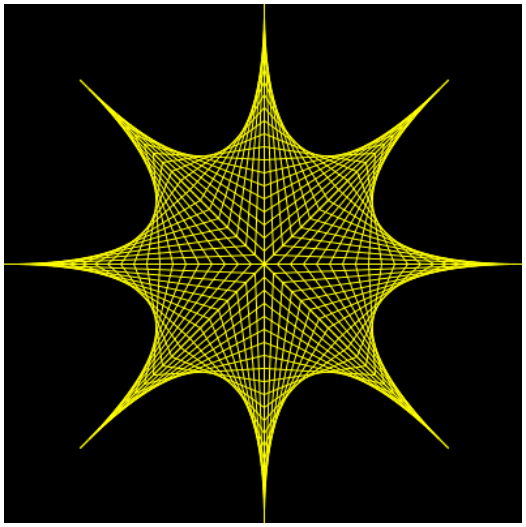
Project 2: Draw Curves from Straight Lines

1 Objective

In this program, you will get in introduction to using objects; we will not create our own just yet. You will also learn about and use typical graphics coordinate systems.

2 Input & Output

This program gathers no input from the user.



2.1 Part 1

- Generate the graphic output shown at right, on a 400x400 DrawingPanel with a black background.
- Alter the figure color to suit your taste.
- Set the increment between lines to something pleasing to the eye; it does not need to match the sample.
- The distance from the center point to the ends of the lines must be the same for the vertical and horizontal axes, and from the center point to the ends of the 45° lines. You will need to think about how to increment evenly to draw properly (e.g., use the Pythagorean Theorem or other geometry).
- For descriptions of the technique and samples of output, see these sites: site1, site2.

2.2 Part 2

- Create an *additional* drawing panel at least 600 x 600. Draw filled-in shapes of *at least two types* (rectangles, circles, etc.) in *at least two colors*. Lines do not count as shapes for this part; add them on top, if you would like, after meeting the requirements.
- Each shape type should be drawn by a function that you write, with parameters indicating where it is to be drawn, how big it is to be, and what color(s) to use for the shape. For example, if you are drawing trees, you might create `drawTree` with parameters for size, location, trunk color, and leaf color. Even if you are only creating simple shapes, create and use one or two such functions.
- Draw many shapes (10+) in an attractive pattern. Do something unique and interesting, perhaps drawing a landscape or creating an eye-catching repeating geometric design. To celebrate your achievements, I will anonymously share with the class some of the most creative work.

3 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 our Course Style Guide.
- Use the DrawingPanel.java provided by your author. Put this in the same folder as your code.
- Use procedural decomposition to break down the program into logical pieces.
- For Part 1, create and use constants for the drawing panel size, line increment, and color. Create additional constants if they help make the code more readable. Only a single loop should be used to draw the figure.

3.1 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 unavoidable, but global constants where they make sense).
- Definite loops.
- Drawing panel and graphics objects.
- Math methods like Math.round, plus trigonometry methods like Math.cos if you would like.

3.2 What You **May Not** Use

- Selection control structures (unless you are doing something special for Part 2). That means no if statements, switch statements, or ternary operators.

4 Submitting Your Work

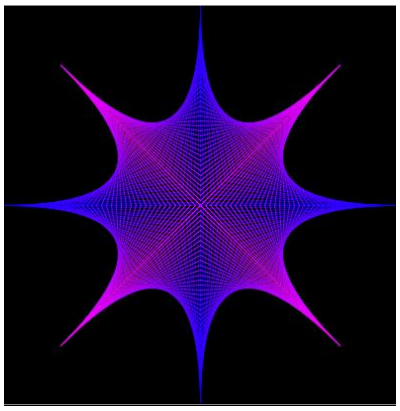
Submit your .java file; there is no need to submit the DrawingPanel.java code.

5 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. It is also linked from inside your BlueJ UI.
- Do not duplicate code if you can avoid doing so; this is a good rule of thumb, now and forever!
- If you would like to watch the figure being drawn, include a Drawing Panel sleep method call.

6 Extra Credit

Instead of using a single color, draw a figure like the one shown at right. The line colors should shift gradually from one color to different color. In the sample, the lines shift from blue to magenta. Add global constants for starting color and ending color; these should work properly with any colors I may specify. **Note: you may not use Graphics2D** or any methods from it; the work can be accomplished with some simple math and creation or modification of Color objects.



7 Grading Matrix

Area	Percent
Part 1 – drawing accuracy	20%
Part 1 – coloring	10%
Part 1 – looping	20%
Part 1 – no globals	10%
Part 1 – general coding	10%
Part 2 – shapes/colors	10%
Part 2 – functions	10%
Documentation and style	10%
Extra credit	5%
Total	**105%**