CS 152 Computer Programming Fundamentals
Project 3: Drawing with Points

Brooke Chenoweth

Spring 2025

In this assignment, you will write a program that uses a class I have provided for you to draw a picture or pattern of points.

1 The `DisplayWindow` class

The `DisplayWindow` class is a small little class. Its only function is to display points. The window keeps track of the last n points that were displayed and draws them as a trace of fading color.

- When creating the display window object, we will specify the number of trailing points and the size in pixels of each point.

  For example, to create a `DisplayWindow` object that will display a trace of fifty points, each five pixels wide, and assign it to a variable named "`window`", you would write following line of code.

  ```
      DisplayWindow window = new DisplayWindow(5, 50);
  ```

- To draw a point at a particular position, you will use the `addPoint` method. The coordinate system we use has $(0, 0)$ at the upper left corner, with x increasing as we go to the right and y increasing as we go down.

  For example, if we were using the `DisplayWindow` variable above, we could draw a point at $(25, 40)$ with the following line of code.

  ```
      window.addPoint(25, 40);
  ```

- I've also provided you with two other methods to tell you how many pixels wide and how tall the display area is.

  Again using the `DisplayWindow` variable above, we can query the display size and assign the values to variables, like so.

  ```
      int width = window.getWidth();
      int height = window.getHeight();
  ```

To use the `DisplayWindow` class, you need to place `DisplayWindow.java` in the `src` directory of your IntelliJ project along with the java file for the program that is using it. I have given you a couple example programs so you can try it out.

2 What you have to do

1. Create a class named `PointPictures`.
2. Create a `DisplayWindow` object in the `main` method.

   You should only create one DisplayWindow object for the entire program.
3. Call the `addPoint` method with a coordinate of $(200, 200)$ and run your program. A point should be drawn at the center of the screen.

   Try calling `addPoint` with other coordinates to see what happens. What happens if you put the method call inside a loop?
4. Now that you've started your program, change it to draw the following in succession.

   - **Box** – Make the display window draw a point that moves around the screen to create a rectangle that is half as wide as it is tall. (For example, if the box is 100 pixels wide, it should be 200 pixels tall.) This means that you will have to modify the indices of the point to move right, then down, then left, and then up. You will need to use several loops for this.

     You may draw the box whereever you like and of whatever size you choose as long has the required proportions, it fits on the window, and the point is shown moving around the perimeter of the rectangle in a continuous path.
   - **Spiral** – I gave you an example of code that drew points on the circumference of a circle, where the angle was changing each iteration of the loop, but the radius was fixed. Consider what happens if you also change the radius. You'll draw a spiral.

     Write a *single loop* that will draw a spiral shape on the same display that you used for the box. Draw the spiral from the outside in, moving in a clockwise direction. Stop spiralling when you reach the center.
   - **Circle on Circle** – Trace a point on the circumference of a circle that rotates around a point on the circumference of another circle. You will only need one loop for this design.

     First, picture a point moving around the circumference of a circle. Use that point as the center of another circle (not necessarily the same size as the stationary circle) and draw the position of a point moving around that circle. The radii of the two circles and the angle moved at each step for the circles should differ from each other for more interesting results.[1]

[1] Certain values result in a degenerate case where you just end up drawing a plain circle in a complicated way. Don't use those.

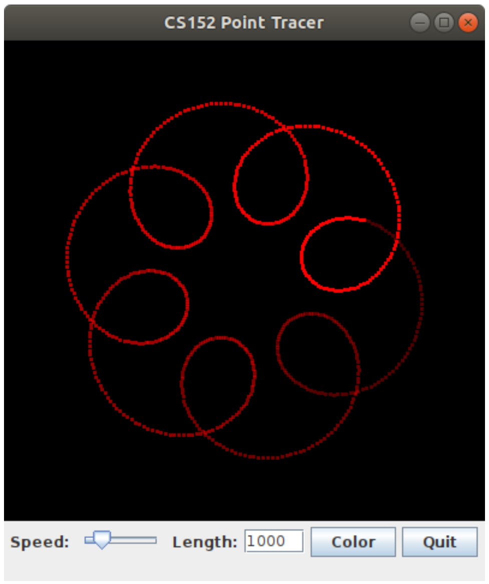
If the explanation in lecture was not enough and you don't understand what we're asking you to draw here, please ask for help in office hours.

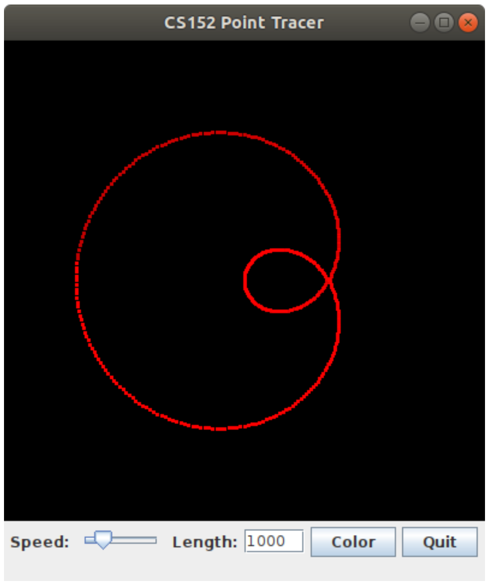
- **Something Fancy** – Come up with your own "fancy" version of a point drawing. The idea should be more complex than the my demos and the first two parts – Can you come up with some nifty pattern?[2] Again, draw this fourth part on the same display as the first three parts.

Make sure that the loops for your first three designs eventually terminate so the graders will be able to see all four parts of your program. The fancy fourth part can run forever if you like.

3 Circle on Circle Examples

Below are two patterns that you might be able to form with the right constants in the circle on circle code.





4 Turning in your assignment

Submit your **PointPictures.java** file to the Project 3 assignment in Canvas. Do not attach `.class` files or any other files. *Only submit one file*, a single program that draws all three required parts.

[2]Past students have drawn smiley faces, stars, flowers, fractals, etc.

5 Grading Rubric (total of 25 points)

[-5 points]:	File submitted to Canvas was not named `PointPictures.java`
[-5 points]:	The code did not compile or compiled with errors or warnings.
[5 points]:	The code adheres to the coding standard specified on the course website.
[5 points]:	Box design
[5 points]:	Spiral design
[5 points]:	Circle on circle design
[5 points]:	Fancy design