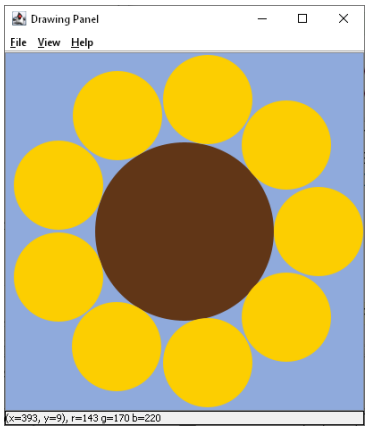
Project 4: Sunflower

1 Objective

This project is the first time you will create your own objects, along with methods, private data, etc. This is not a complete working system with user interaction; it is just a chance for you to create classes, then instantiate and test them[1]. You will create two supplier classes, one of which uses the other ("has-a" relationship).

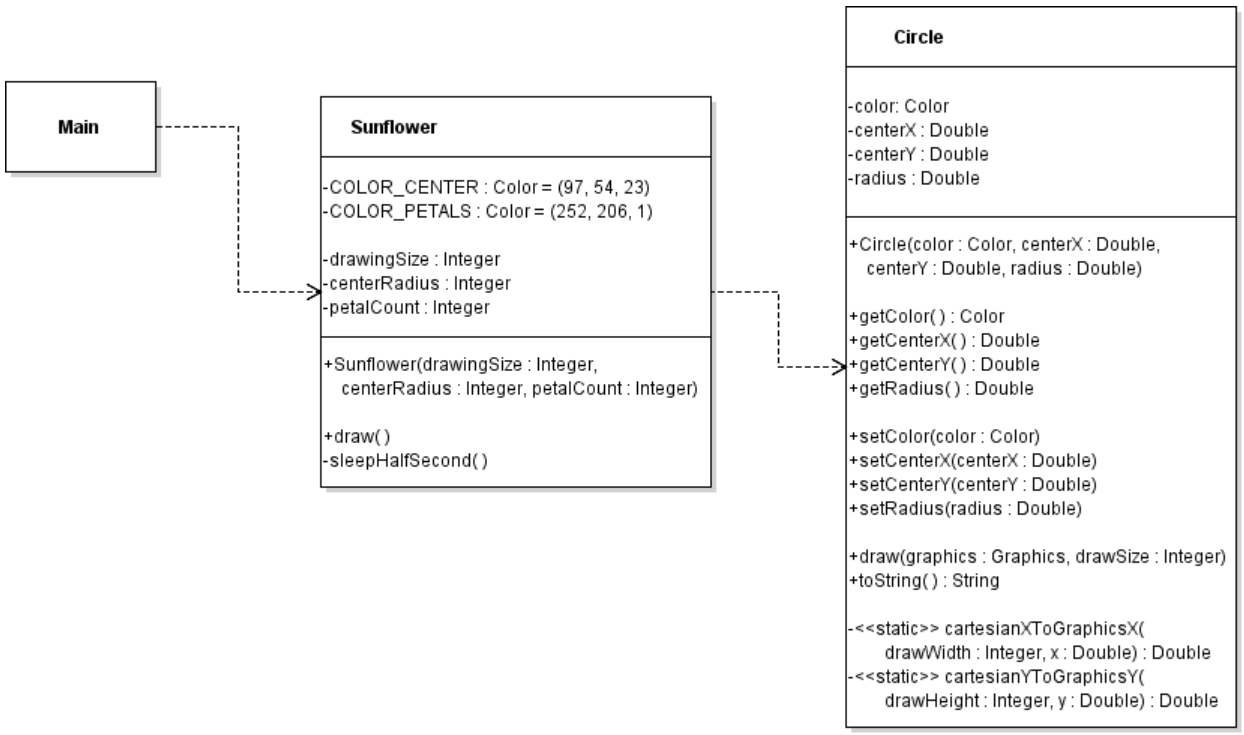


2 Output

Sample output from the project is shown at right. In this sample, the drawing area is 400 pixels in width and height. The center of the flower has a radius of 100 pixels. There are 9 petals. The client will be able to specify all three options when instantiating the Sunflower.

3 Classes to Create

Here is the UML Class Diagram depicting classes you will create and how they relate to each other. You supply the data section of both classes. Please pay attention to notation (including parameter and return types); there is a good deal of information provided.



[1] You are mostly writing Supplier code here, not Client code. Main serves as a simple demonstration of what clients might do with your classes, however.

4 More Details about Classes

4.1 Circle

Stores information necessary to draw a circle and provides a method that draws the shape.

Think in cartesian coordinates for the data; translate to graphics coordinates using the static translation methods when you are ready to draw. Note that the draw method requires a Graphics reference as well as the size of the drawing canvas, so it can position its data properly and call the appropriate Graphics commands to set colors, draw shapes, etc.

The class's toString method should produce something like this, useful for debugging purposes:

```
Circle centered at (0.0000, 0.0000) with radius 100.0000 and color java.awt.Color[r=97,g=54,b=23]
```

4.2 Sunflower

Via constructor parameters, clients specify the size of the drawing area (square, always), the radius of the flower's center, and the petal count. Observe these when creating the DrawingPanel and drawing the flower. Note: this class will not *draw* any Circles; it will *create* them and ask them to *draw themselves*.

Create and draw the center first. The petals should be evenly arranged around the center, with the petals exactly tangential to the center. The first petal should be drawn directly to the right of the center. Petal radius is calculated at 50% of the center's radius. You'll need to put your math knowledge to use, determining exactly where the center of each child will be, regardless of how many petals are requested.

Colors (RGB):

- Sky: 143, 170, 220
- Center: 97, 54,23
- Petals: 252, 206, 1

In class and/or office hours, let's discuss how to write *sleepHalfSecond*, which will be used to pause between drawing petals so you can see them appear in order, starting from the position directly to the right of the center, moving counterclockwise around the center.

5 Main

Main should create and draw a Sunflower. Pass arguments that create an attractive result, e.g., the ones shown in Section 2, "Output," which are 400 (canvas size), 100 (center radius), and 9 (petals).

6 Constraints and Assumptions

- Mark each instance variable and each method as either public or private (and use no other modifier). Follow the UML Class Diagram; it tells you exactly which modifier to use.
- Look carefully at the parameter data types and the return data types; they give you significant clues.
- Throw an IllegalArgumentException (with an informative message, always) if any of these preconditions are violated:
  - Object references are null (e.g., color).
  - Radius is negative.
  - Draw sizes are less than 1 pixel.

7 Style and JavaDoc

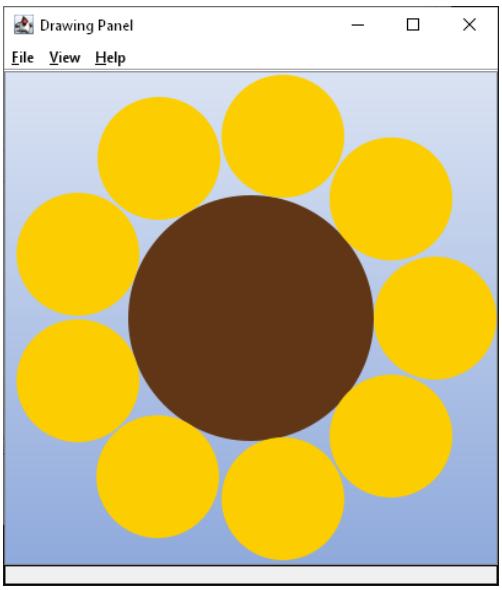
Follow our Course Style Guide found in Canvas. Write JavaDoc for the Circle class.

8 Testing

Write *JUnit5* (see slides) unit tests for the Circle class. Test constructors, all methods, and preconditions. And do not forget test code is *still code* and can have bugs, so be suspicious if everything passes miraculously the first time, i.e., test that you can induce failures and get the appropriate failure messages.

9 Extra Credit

Create two constants, COLOR_BACKGROUND_TOP (218, 227, 243) and COLOR_BACKGROUND_BOTTOM (143, 170, 220). Write a method called drawGradientBackground. Using only setColor, drawLine, and math (no Graphics2D methods!), draw a gradient background instead of a solid one.



10 Submitting Your Work

If using BlueJ, use the IDE to create a .jar file. Be sure and specify "include source." Submit the .jar file. Or, using your OS, zip up your source files and submit a .zip file.

11 Hints & Reminders

- Do not duplicate code; avoid this where possible.
- Everything you write in this project (except main) is Supplier code. In supplier code, you should not ask questions of the user nor print anything on the terminal/console.

12 Grading Matrix

Achievement	**Max**
Circle class	0%
Constructor	5%
Set methods	5%
Get methods	5%
Minimal code duplication	5%
Draw method	15%
Preconditions	5%
Sunflower class	0%
Constructor	5%
Draw method	20%
sleepHalfSecond method	5%
Preconditions	5%
Main class	5%
Testing of Circle class	10%
Style and Documentation	10%
Extra credit (gradient)	5%
Total	**105%**