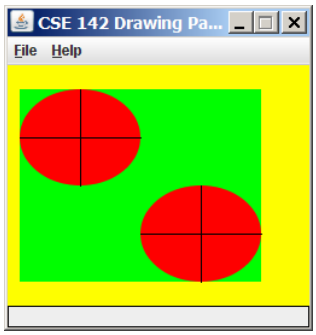
CSE 142, Spring 2007
Programming Assignment #3: Doodle and Circles (20 points)
Due: Tuesday, April 17, 4:00 PM

Program Description:

This assignment will give you practice with parameters and graphics. This assignment has two parts; turn in two Java files named `Doodle.java` and `Circles.java`. To compile and run this assignment, you must download the file `DrawingPanel.java` from the Assignments section of the class web page and save it in the same folder as your code. You should not turn in `DrawingPanel.java`.

Part A: Doodle (4 points)

For the first part of this assignment, turn in a file `Doodle.java` that draws a figure using the `DrawingPanel` provided in class. You may draw *any figure you like* that is at least 100 x 100 pixels, contains at least three shapes, uses at least two distinct colors, is not highly similar to your figure for Part B. Be creative! Your doodle does not have to be a static image—consider animating it! After the assignment is turned in, we will anonymously post student figures on the course web site as part of a just-for-fun Doodle voting contest.

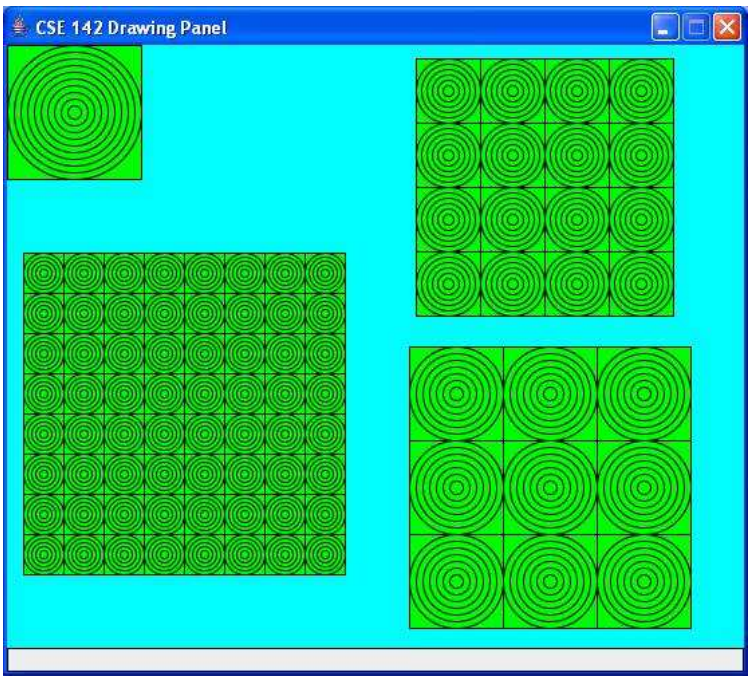


If you do not want to create a unique figure, at left is a default figure you may draw for Part A. If your Doodle program draws this figure, you will receive full credit for Part A. This figure contains a `DrawingPanel` of size 250 x 200 with a yellow background, a green rectangle with the top-left corner at (10, 20) and size 200 x 160, red ovals with top-left corners at (10, 20) and (110, 100) of size 100 x 80, and black lines from (10, 60) to (110, 60), from (60, 20) to (60, 100), from (110, 140) to (210, 140), and from (160, 100) to (160, 180).

You may optionally wish to try using parameterized methods to capture the repetition in the figure. However, your score for Part A will be based solely on its external correctness as defined above; it will not be graded for internal correctness or style.

Part B: Circles (16 points)

The second part of this assignment asks you to turn in a file `Circles.java` that draws a specific complex figure that consists of several grids of squares at various locations and sizes, containing a varying number of concentric circles. Your program should exactly reproduce the image shown here.



The images in this document were taken on Windows; if you use another platform, the window may look slightly different.

The program should produce graphical output only; no text output should be printed.

The `DrawingPanel` provides functionality to compare your image against the expected output file posted on the course web site. Detailed specifications of the appearance of the figure are on the next page.

Implementation Guidelines (Part B):

The overall drawing panel is size 550 x 450. Its background color is cyan. The panel contains four figures of squares with concentric circles. The concentric circles are drawn in black against a green background. The four figures on your drawing panel should have the following properties:

Description	(x, y) position	number of rows and columns	size of each subfigure, in pixels	number of concentric circles in each subfigure
top-left	(0, 0)	single subfigure	100 x 100	10
bottom-left	(12, 155)	8 x 8	30 x 30	5
top-right	(305, 10)	4 x 4	48 x 48	6
bottom-right	(300, 225)	3 x 3	70 x 70	7

Stylistic Guidelines (Part B):

To receive full credit on Part B, you should write static methods that use a great deal of parameter-passing and some numeric computations. You are required to have **two useful static methods besides `main`**. One static method draws a grid each time it is called. You will call this method three times from `main` to produce the three grids in the overall figure. It will need several parameters to be flexible enough to draw each of these grids. The key point is that a single method can be called three times to produce the three grids. If all of the grid-drawing code were placed into the preceding method, it would be quite large. Therefore you should decide on a second useful method to add to the program that offloads a nontrivial portion of the drawing logic.

In grading, we will require these two static methods: one for a subfigure with just one square and its concentric circles and one for producing a grid of such figures that is called three different times to produce the three grids. Use proper indentation and formatting to make your program readable. Give meaningful names to methods and variables and follow Java's naming standards. Localize variables whenever possible; that is, declare them in the smallest scope in which they are needed. You will not be using class constants. Include a comment at the beginning of your program with a description of the program and include a comment at the start of each method explaining its behavior.

For this assignment you are limited to the language features in Chapters 1 through 3; you are not allowed to use more advanced Java features to solve the problem. For reference, our solution is 54 lines long including comments.

How to Get Started (Part B):

This program does not require a lot of lines of code, but the computations are not simple. You might very well find yourself overwhelmed with the amount of detail you have to handle all at once. The techniques of decomposition and iterative enhancement described in Chapter 1 will help you in this situation.

Instead of trying to correctly implement such a complex program all at once, you should start with a smaller piece of the problem. In particular, you should write a static method that draws one square with concentric circles inside of it. That subfigure is repeated throughout this image, so it makes sense to have a separate method for producing it. Start with the subfigure in the upper-left part of the screen.

Your first version of this method could draw the specific subfigure in the upper-left corner (always drawing it in that position, always making it 100 pixels wide, always having 10 concentric circles), but you'll want to generalize this with the use of value parameters. You should be able to call it with different sizes, different locations and different numbers of concentric circles. Once you have completed the static method that produces one of these subfigures, write another static method that produces a square grid of these subfigures.

Write the code incrementally and make gradual improvements, making sure to test the code thoroughly at each step. Eventually, you will have a parameterized method that allows different grid sizes, (x, y) locations, and so on.

By the way, the drawing commands we are using are defined in terms of integers, which leaves open the possibility that things won't divide up evenly. You don't have to worry about this possibility. In particular, you may assume that the subfigure size will always be a multiple of twice the number of concentric circles you are supposed to draw (e.g., 10 circles in a figure 100 wide, 5 circles in a figure 30 wide, 6 circles in a figure 48 wide, 7 circles in a figure 70 wide).