

Programming Assignment #3: Doodle (25 points)

Due: Wednesday, October 28, 2009, 11:30 PM

Program Description:

This assignment covers parameters and graphics. Turn in Java files named `Doodle.java` and `Circles.java`.

To compile and run this assignment, you must download the file `DrawingPanel.java` from the class web page and save it in the same folder as your code. Do not turn in `DrawingPanel.java`.

This assignment is a pair assignment. This should help you catch mistakes as you make them and get a feel for what working in a collaborative environment is like. Both group members must be involved in both planning and typing. **Each of you must turn in your own doodle.** I'd recommend starting with the structured portion of this assignment and then going crazy with the doodle!

Implementation plan (5 points):

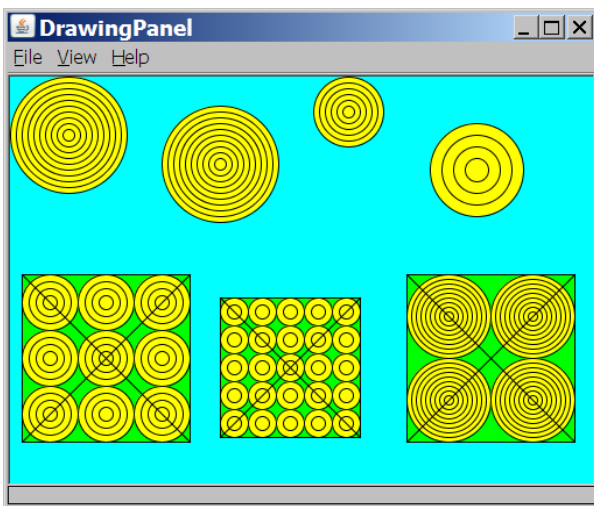
You will complete this assignment as a pair. Before you go to the computers, you will have to show me an implementation plan. This will have to detail which methods you will write, what parameters they will accept as well as pseudocode for them. Include as much detail as you can, including how you will split up work so that both partners are engaged and contributing.

Part A: Doodle (5 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 your own work, and is not highly similar to your figure for Part B.

Be creative! You may want to do this after you have completed Part B and decide how complex you want to make it based on the time you have. I'll post cool ones on the website.

Part B: Circles (15 points):



The second part of this assignment asks you to turn in a file named `Circles.java` that draws a specific figure of grids of concentric circles. Your program should exactly reproduce the image at left.

The Part B image has several levels of structure. There is a basic "subfigure" that occurs throughout, containing concentric circles inside it. The subfigure is repeated to form larger grids.

The overall drawing panel is size **500 x 350**. Its background is cyan. The rectangular area behind the grids is green, and the background of the circles is yellow. The rectangles and circles are outlined in black. Each grid also has a pair of lines drawn across it in an "X" pattern.

The figures on the panel should have the following properties.

| Description | (x, y) position | size of subfigure | circles per subfigure | rows/cols |
|---------------|-----------------|-------------------|-----------------------|-----------|
| top-left | (0, 0) | 100 x 100 | 10 | N/A |
| top-middle 1 | (130, 25) | 100 x 100 | 10 | N/A |
| top-middle 2 | (260, 0) | 60 x 60 | 6 | N/A |
| top-right | (360, 40) | 80 x 80 | 4 | N/A |
| bottom-left | (10, 170) | 48 x 48 | 4 | 3 x 3 |
| bottom-middle | (180, 190) | 24 x 24 | 2 | 5 x 5 |
| bottom-right | (340, 170) | 72 x 72 | 9 | 2 x 2 |

Development Strategy (How to Get Started):

This program does not require as many lines of code as past ones (my solution is under 65 lines). But the numeric computations and parameters are not simple. You might be overwhelmed with the amount of detail you have to handle all at once. As famous computer scientist Brian Kernighan once said, "Controlling complexity is the essence of computer programming." To make things easier, begin with a smaller piece of the problem.



It may help you to compute a value that we'll call "**the gap**," or the distance between each neighboring pair of concentric circles in a subfigure. The 100x100 top-left subfigure has 10 circles, and each circle has a gap of 5 pixels from any others (a total of 20 gaps in each direction). You could store the gap in a variable and use it in your subfigure drawing code. Each grid figure uses a different gap value for its subfigures, based on the parameters.

Write your code incrementally, repeatedly making small improvements. Start out by having your first method draw only the subfigure in the upper-left part of the screen. Then generalize it by **adding parameters one at a time**. For example, add parameters to change the subfigure's x/y position. Test the code by passing different values. Once one parameter works, move on to the next.

Style Guidelines:

We require at least two methods: one to draw the circle subfigure and one to draw a grid of them. You may use additional methods if you like. You may receive a deduction if your methods accept too many parameters or unnecessary parameters.

Give meaningful names to methods, variables, and parameters, and properly indent your code. Limit the lengths of your lines to fewer than 100 characters. Include meaningful comment headers at the top of your program and **at the start of each method**.