Assignment 3F - Drawing Loops

## Part I

Import RepeatingPattern.java into your new project. This is the same code you used in assignment 2I to draw things on the screen. Make your changes starting on line 9. See the picture below. Create that pattern on the screen; the circles are 10 pixels high and 10 pixels wide. Use **for** loops instead of writing separate lines of java for each circle. Do not draw circles past the edge of the screen where they can’t be seen; your loop must take care of stopping itself at the end of the screen.

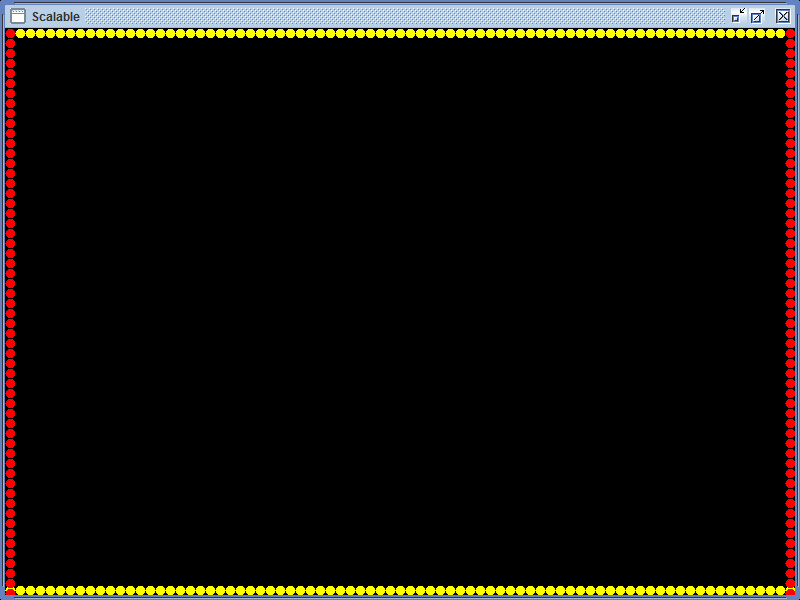
Recap of commands available:

getWidth() gives you the width of the screen in pixels

getHeight() gives you the height of the screen in pixels

g.fillOval(x, y, width, height) draws a circle with upper left-hand corner at position x, y

g.setColor(Color.YELLOW) sets the pen drawing color



**(OVER)**

## Part II

In project 2I, you drew some shapes on a Java GUI. In this folder is another Java GUI – ColorBars.java. Import that into eclipse into a new project. It has some buttons on the screen. Try it out – run it and click on both buttons to see what happens.

Look at the code where it says **actionPerformed** starting on line 17. This code block runs in response to the user clicking either of the buttons at the bottom of the screen. When the user clicks the *draw* button, the **if** executes. When the user clicks the *clear* button, the **else** executes. You are going to complete the bodies of the **if** and the **else.** You are not responsible for knowing about all the surrounding framework code (although comments are there to describe it); you’re only responsible for the code I'm asking you to add.

### Colors

Currently, when the user presses the button for drawing, the program creates a new color object using the **new Color(150, 0, 0)** statement. There are three integer parameters that get passed to the Color object. The first represents the amount of red in the new color, the second is the amount of green, and the third is the amount of blue. The easy mnemonic to remember this is RGB or Red, Green, Blue. Each color parameter can be an integer from 0 to 255. Zero for red means “no red”; 255 for red means “all red”, and so on. Therefore there are 2563, or 16,777,216 (16 million) possible colors. (You are not responsible for Color on tests or the AP test)

Examples (try some out):

* new Color(0,255,0) would be no red, all green, and no blue.
* new Color(255,0,255) would be a combination of all red and blue, but no green

### Clearing the screen

When the clear button is pressed, the program draws a black square on top of the red square that the draw button makes. Actually, instead of just drawing in black, it retrieves the background color of the window and draws a square in that color to erase the red one. The visual effect to the user is that the screen is cleared.

### Your Assignment

1. Change the program so that when the user clicks the draw button, the program **repeatedly** draws 256 thin rectangles as follows (use a for loop):
   * The size of each rectangle will be 100 pixels wide by 2 pixels tall
   * The position of the first rectangle should be at horizontal pixel 5 and vertical pixel 10
   * Each successive rectangle should start at the pixel directly underneath the previous rectangle. Each rectangle is two pixels tall, so this means you will have to skip down two pixels each time.
   * The color of the first rectangle will be black: red=0, green=0, blue=0
   * The color of each successive rectangle will have the red component increased by 1. The green and blue components will remain at zero.
   * Test this much and get it working
2. Now, add similar patches for green and blue. The green patch should be to the right of the red patch far enough so that there is a gap between them. The blue should be right of the green, far enough so that there is a gap between them.
3. Now add a patch that starts at 0 0 0 for RGB and increases each component at the same time up to 255 255 255. Put it to the right of the other patches.
4. Now try one more patch. Make it so that each successive square is a random color in the full spectrum range of 0 to 255 for each of red, green, and blue. Put it to the right of the other patches. Test to see what happens when you click the draw button repeatedly without clicking clear between them. **NOTE: the random numbers you generate must be exactly in the range of 0 to 255. You will lose a point and not be able to recover the point if it gives the range 1 to 255 or 0 to 254 or something other than 0 to 255. Please ask if you are not sure.**
5. When the user clicks the button to clear the screen, make sure all of the colored rectangles get erased. It is more efficient to clear the screen with one large rectangle rather than drawing many small rectangles.
6. Now slow the program down. Slowing down and controlling speed of execution is the foundation of computer animation. Since computers are so fast, you basically need to cause the execution to pause for a short period of time to give the illusion of animation. This statement will cause java execution to sleep for 500 milliseconds (one-half of a second):

try {Thread.sleep(500);}

catch (InterruptedException ioe) {}

You are not responsible for this statement on a test or the AP test. Change your program so that the color swatches appear to be drawn gradually down the screen.

Please don’t make it so slow that it takes me forever to test it and grade it, but it should be visibly animated.