This is a regular project for **AP students** and an extra credit project for **Honors students**.

# Setup

Create a new project called 2I *yourlastname* in Eclipse. Import the pre-made java file for this project from my outbox.

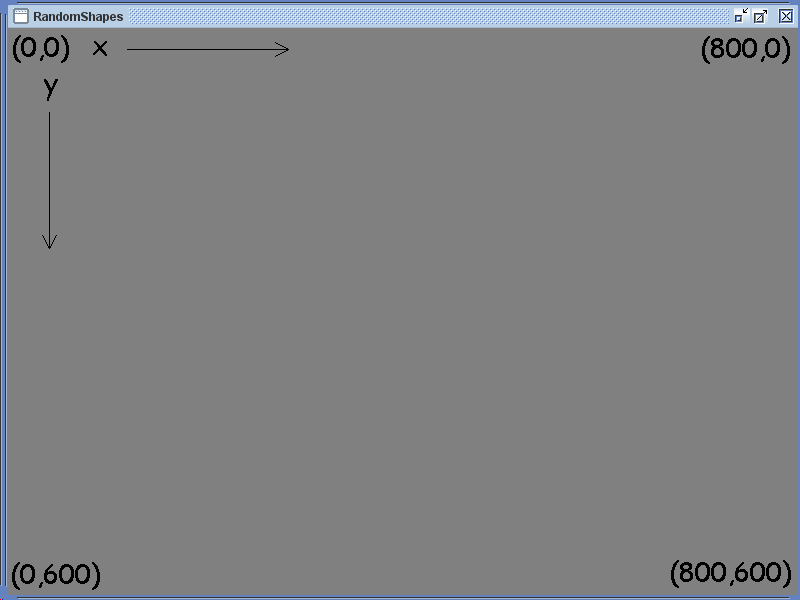
Look at ScalableShapes and run it. The only thing it does is pop up a black window. You are going to make changes starting on line 9.

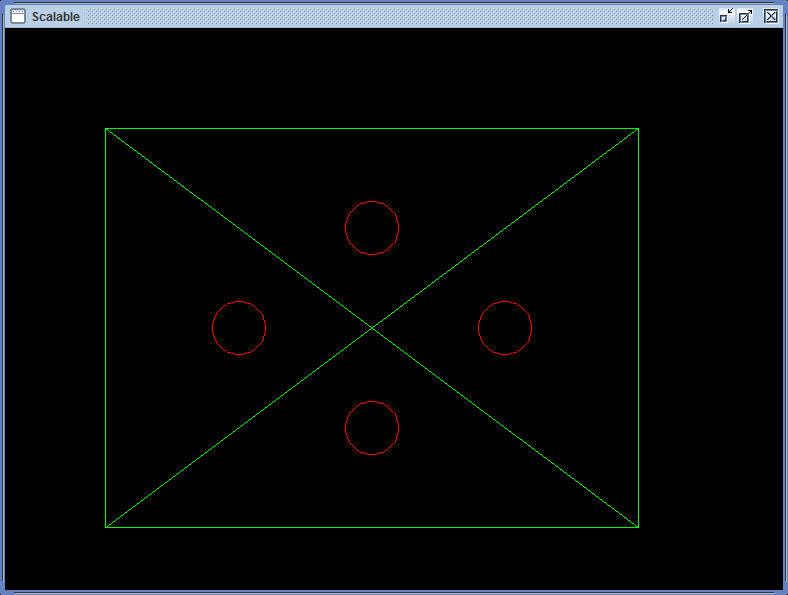
1. Notice that I am giving you a Graphics object variable called **g** on line 6. A graphics object is used for drawing shapes and colors. It is not included on the AP test (or my test); this is just for fun (but I’m counting it for points). Here are a few of the methods you can call on a Graphics object:

|  |  |  |
| --- | --- | --- |
| **Data type of result (void means there is no result)** | **Method and Parameters** | **What it does** |
| void | setColor(Color c) | Changes the drawing color |
| void | drawLine(int x1, int y1, int x2, int y2) | Draws a line from (x1,y1) to (x2,y2) |
| void | drawRect(int x, int y, int width, int height) | Draws a rectangle with upper-left-corner at (x, y) and size specified by width and height. |
| void | fillRect(int x, int y, int width, int height) | Draws a solid rectangle with  upper-left-corner at (x, y) and size specified by width and height. |
| void | drawOval(int x, int y, int width, int height) | Draws an oval |
| void | fillOval(int x, int y, int width, int height) | Draws a solid oval |

Try one out. For example try to draw a rectangle at 100,200 with width 300 and height 400.

Here’s how the pixel layout of the screen works. The origin is the upper-left-hand corner:



1. Change what you did in step 1 to draw a rectangle at (100, 100), with a random width between 0 and 600, and height equal to ¾ of the width (go ahead and let the decimals get chopped off). Your code will get executed every time Windows needs to re-paint your screen. For example, run the program and drag the lower corner of the window to resize it. Notice how the size of the rectangle changes. That’s because your code for making a random sized rectangle is executing every time Windows signals a re-paint.
2. Now draw a line from the upper left-hand corner to the lower right-hand corner *of the rectangle* (not the window, but the rectangle). This will form a big “X” inside your rectangle.
3. Now draw the other diagonal going to the other two corners of the rectangle.
4. Now draw an oval.
   * The width and height of the oval will be the width of the rectangle divided by 10.
   * The x coordinate should be 100, plus half the width of the rectangle, minus half the width of the oval.
   * The y coordinate should be 100, plus one-quarter the height of the rectangle, minus half the height of the oval.
5. Draw another oval, same size as the last one.
   * The x coordinate should be 100, plus one-quarter the width of the rectangle, minus one half the width of the oval
   * The y coordinate should be 100, plus half the rectangles height, minus half the oval’s size.
6. Draw two more ovals to complete the pattern (see below for an example). Be sure to change the window size to see that everything remains relative.
7. Now, to enhance the pattern, change the color of the ovals; just before drawing the ovals, add a g.setColor(Color.YELLOW).

There are literally over 16 million colors (2563). Instead of picking from those, there are a handful of pre-built ones that you can use: BLACK, BLUE, CYAN, DARK\_GRAY, GRAY, GREEN, LIGHT\_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW (all with ***Color.*** In front of them).

1. That’s it; you made a nice scalable, colorful pattern.