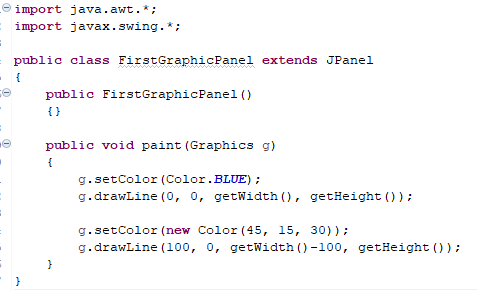
# Introduction to Graphics Part 2

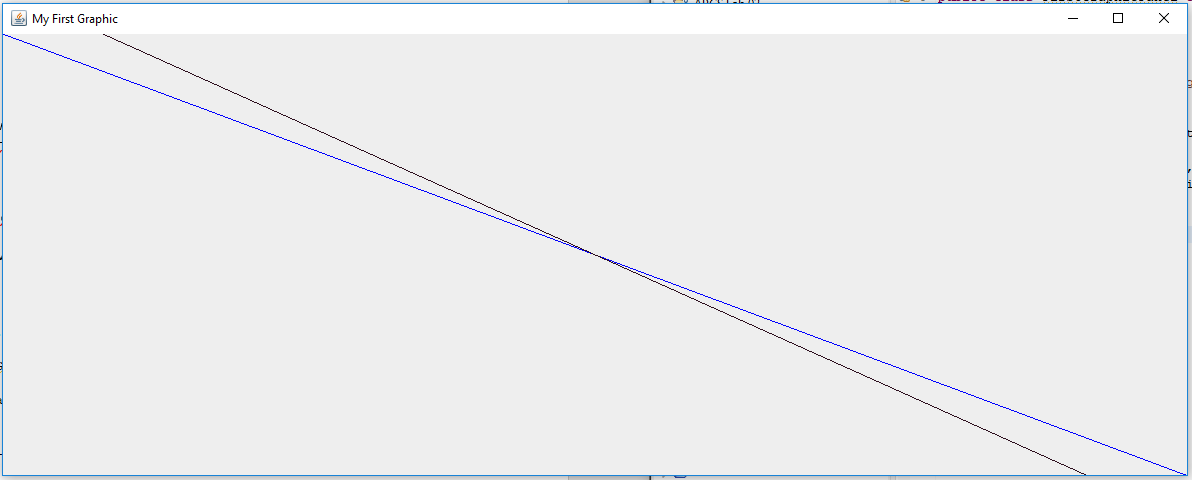
The programmer will now explore some of the methods available to draw graphics by continuing to build off of the GraphicsIntro Project.

## setColor(Color c)

setColor is a method that changes what the color the writing will be. The programmer can set the color to a defined color like Color.blue or Color.red. The programmer can also create a new color based on RGB values. Each value is between [0,255] inclusively. The first value is the red hue, the next value is the green hue, and the third value is the blue hue. In the program below, I am choosing not to use super.paint() because it is not an event driven program.



The code would produce the following window.



## drawLine(int x1, int y1, int x2, int y2)

The drawLine method will draw a line from (x1, y1) to (x2,y2). This is not a Cartesian Plane. The x value is the columns from left to right and the y value is the rows from top to bottom. This means (0,0) is in the upper left corner.

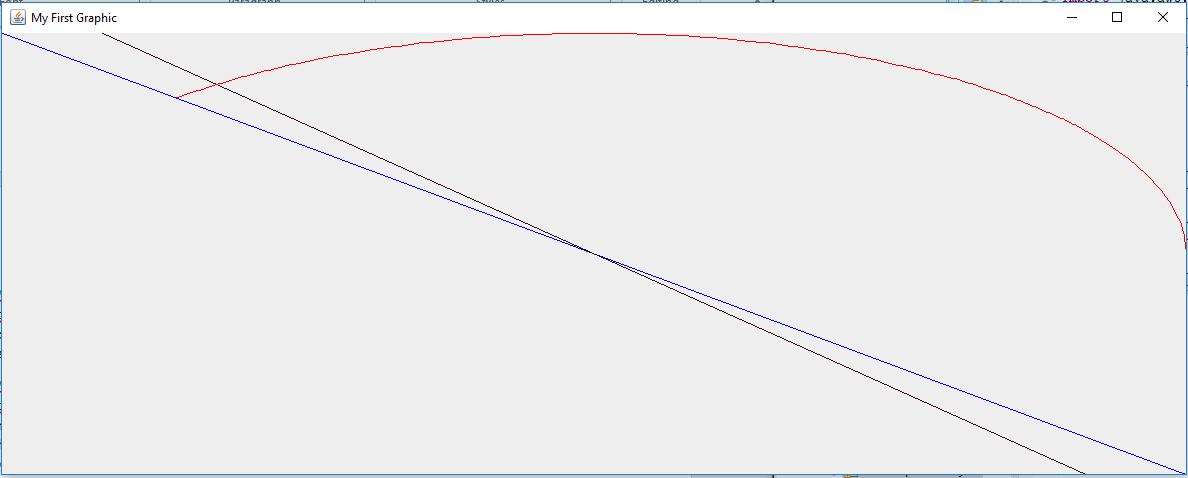
## drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)

The drawArc is not as intuitive as the previous methods. The x and y coordinate is the upper left corner of a rectangle that would incase an oval that could be drawn within it. The width and height is the width and height of that rectangle which would enclose said oval. The start angle is where the drawing begans, based on the unit circle. This means that zero degrees would be the equivalent of starting at 3:00 on analog clock and the drawing will continue in a counter clockwise fashion (or widdershins if you’re Scottish or a fan of Terry Prachett’s Discworld). The code:

g.setColor(Color.***RED***);

g.drawArc(0, 0, getWidth(), getHeight(), 0, 135);

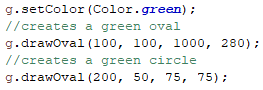
would create the following



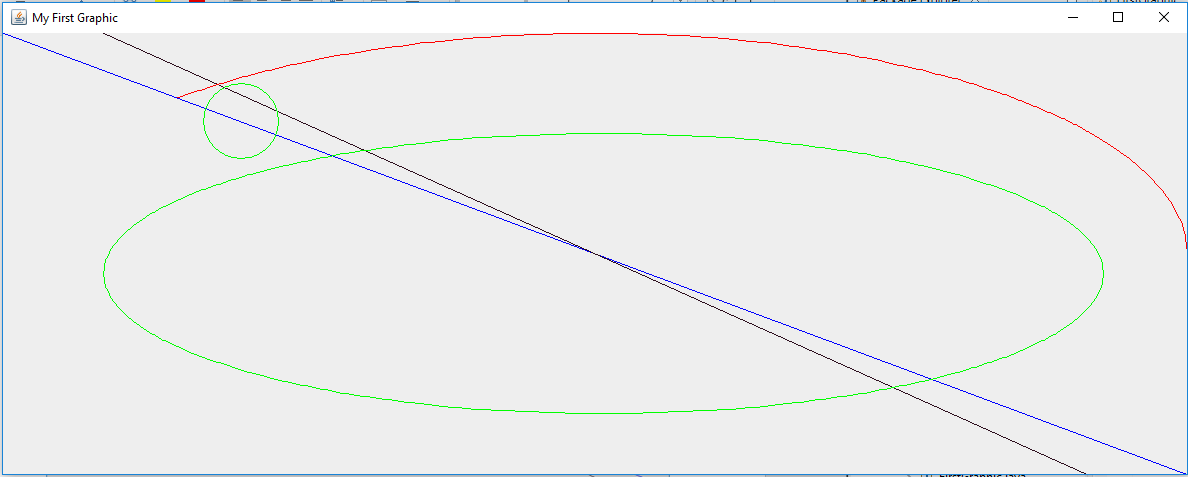
There is also a method called fillArc with the same parameter list that will create an arc that is filled in with the color of the drawing. The arc is combined with the radius of the oval and center of the arc to create a shape.

## drawOval(int x, int y, int width, int height)

The drawOval will draw an complete oval based on the width and height of the rectangle that could enclose it. The x and y point is the upper left corner of the circumscribing rectangle. A perfect circle can be drawn using width and height to describe a circumscribing square. The code:



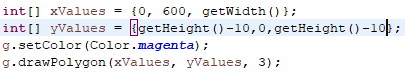
Would create:



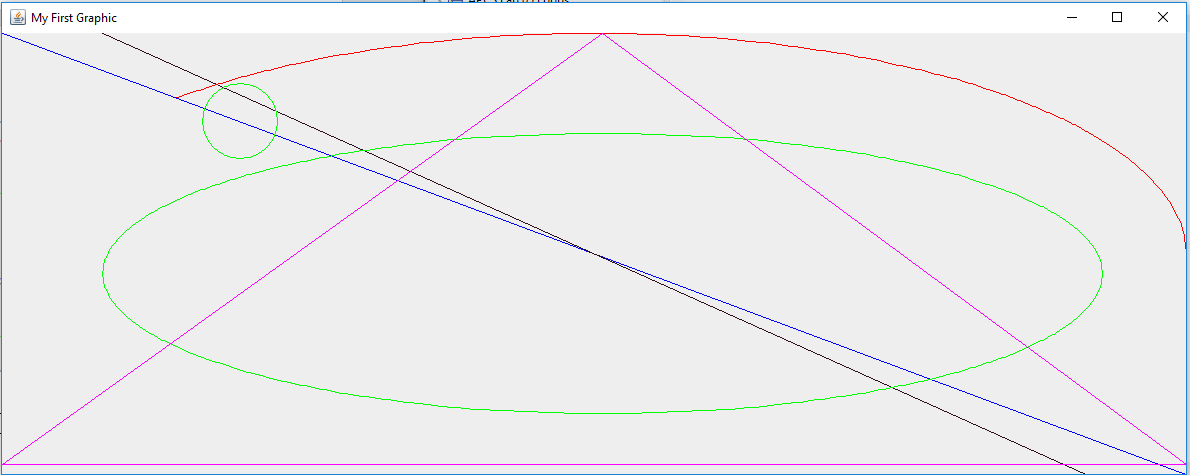
There is also a fillOval method with the same parameters that color in the oval with the same color.

## drawPolygon(int[] xPoints, int[] yPoints, int nPoints)

The drawPolygon method will draw a many pointed polygon. The method receives two arrays of size n and an integer indicating the size of the arrays. The arrays are in parallel. The last set of points will connect to the first set of points. The code



would result in

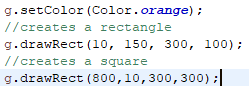


There is also a method called drawPolyline(int[], int[], int) which does the same thing but does not connect the last and first points. There is also a Polygon class that does the same thing as the parallel arrays. The method that uses the Polygon class is drawPolygon(Polygon p).

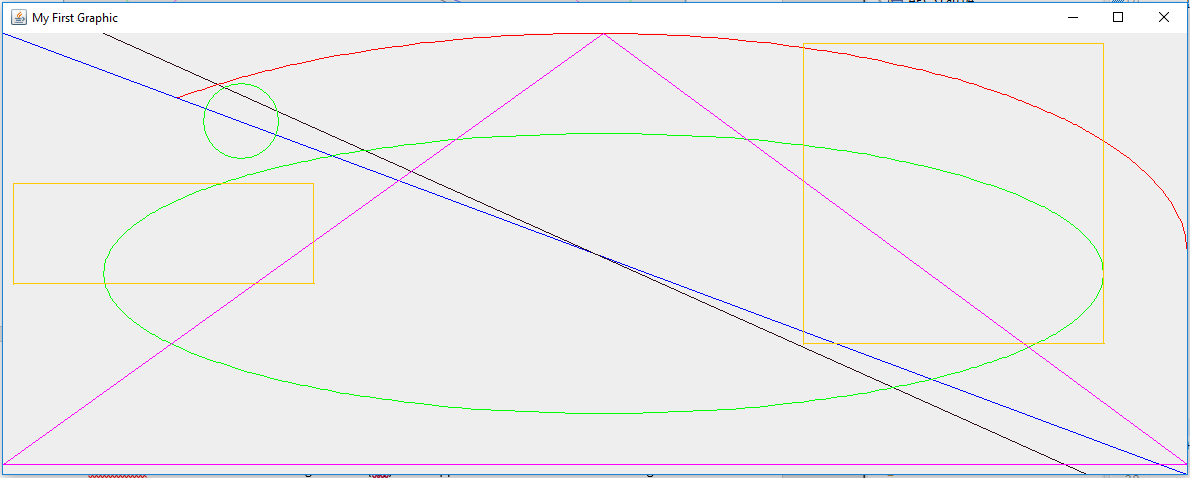
There are also respective fillPolygon methods with the same parameters that will fill in the shape created.

## drawRect(int x, int y, int width, int height)

The drawRect method will draw a rectangle where (x,y) is the upper left hand corner of the rectangle and width and height describe the dimensions of the rectangle. If the programmer would like to draw a square, then they would use the same value for both width and height. The code



would create the following window:



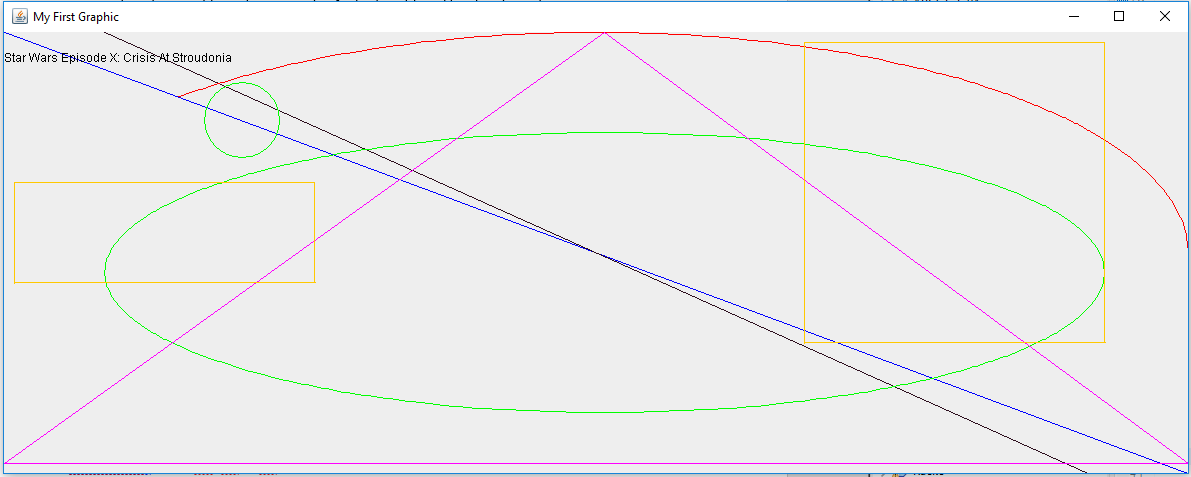
There is also a fillRect method with the same methods that will fill in the shape created.

## drawString(String str, int x, int y)

The drawstring method will print the String provided at (x,y). In this case, the (x,y) coordinate is the the lower left of where the words would appear. The programmer could also change the Font using setFont(Font font) where Font is an object class. The code



Would produce the following:



## Other methods and other Graphics classes

This is just the Graphics class and it is an older version of it. If the programmer looks at the API, they will see other methods in the Graphics class as well as other types of Graphics classes. The programmer is free to try and Graphics classes they feel like and to experiment with those classes.