Statically Drawing Shapes

Drawing on a graphics object

Documentation:

https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html

https://docs.oracle.com/javase/7/docs/api/java/awt/Color.html

Some Graphics Class Methods:

clearRect(int x, int y, int width, int height)
Clears the rectangle with the top left corner (x,y) with the given width and height so that it returns to the background color.

drawChars(char[] data, int offset, int length, int x, int y)
Draws the text given by the specified character array, using this graphics context's current font and color.

drawLine(int x1, int y1, int x2, int y2)
Draws a line on this graphics object from location (x1, y1) to (x2, y2).

drawOval(int x, int y, int width, int height)
Draws the outline of an oval with the top left corner (x,y) with the given width and height.

drawRect(int x, int y, int width, int height)
Draws the outline of a rectangle with the top left corner (x,y) with the given width and height.

fillOval(int x, int y, int width, int height)
Fills an oval with the top left corner (x,y) with the given width and height with the current color.

fillRect(int x, int y, int width, int height)
Fills a rectangle with the top left corner (x,y) with the given width and height.

setColor(Color c)
Sets the color to c.

Color Class Constants, Constructor

Constants: BLACK, black, BLUE, blue, CYAN, cyan, GREEN, green, ORANGE, orange, PINK, pink, RED, red, WHITE, white, YELLOW, yellow

Color(int r, int g, int b)
Creates an opaque sRGB color with the specified red, green, and blue values in the range (0 - 255).

PaintStuff Framework

Let's take a basic look at the framework necessary to open a window where we draw objects on it using a Graphics object.

A JFrame is a window in Java Swing. First, we must create a window and set some parameters for it. Here is the example from PaintStuffFramework.java:

```
JFrame frame = new JFrame("Homework Problem 1");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(500, 500);
```

The first line of code creates a JFrame object, specifying its title. The second line of code simply indicates that the frame will disappear when the user clicks on the x at the top right. The third line of code sets the initial size of the JFrame.

From here, we create a new JComponent (which is called ContentViewSolution in our example), and add this to our frame. Then we set a couple more components of our frame:

```
ContentViewSolution view = new ContentViewSolution();
frame.add(view);
frame.setResizable(false);
frame.setVisible(true);
```

JComponent (ContentViewSolution)

This object will just contain an object of the class we create that will specify what gets drawn on the frame. A JComponent has a special method paintComponent that takes in a graphics object g. In this method, we'll simply call the draw method, passing it the graphics object to do all of our actual drawing. Here is the ContentViewSolution class in its entirety:

```
class ContentViewSolution extends JComponent {

        StudentWorkSolution student;

        public ContentViewSolution() {
                student = new StudentWorkSolution();
                student.initialize();
        }

        public void paintComponent(Graphics g) {
                student.draw(g);
        }

}
```

Class with our solution: StudentWorkSolution

Though no instance variables are required here, in this example we just have two colors, one for characters and one for the background. In our default constructor, we set these to green and white, respectively:

```
class StudentWorkSolution {

        Color characterColor, backgroundColor;

        public StudentWorkSolution() {
                characterColor = Color.green;
                backgroundColor = Color.white;
        }
}
```

We have an initialize method (which is empty here) which gets called from the ContentViewSolution constructor:

```
public void initialize() {
}
```

Finally, the draw method was already called in the paintComponent method. Thus, all of our actual works for drawing goes in the draw method of the StudentWork Solution class. Here we can set the color and draw objects and change our color as we please.

Here is a simple example that fills the background white and then paints a green circle:

```
public void draw(Graphics g) {

        g.setColor(this.backgroundColor);
        g.fillRect(0, 0, 500, 500);

        g.setColor(this.characterColor);
        g.fillOval(200, 200, 100, 100);
}
```

This circle will be right in the middle of the frame, since it will go from (200, 200) to (300, 300) and the size of the frame is 500 x 500.

In the same manner we can test several of the other graphics methods as follows:

We draw a magenta rectangle:

```
g.setColor(Color.MAGENTA);
g.fillRect(10, 100, 80, 20);
```

This will appear to the top left, 10 pixels from the left end and 100 pixels from the top.

Here we write some text:

```
g.setColor(Color.BLACK);
g.setFont(new Font("TIMESNEWROMAN", Font.PLAIN, 40));
g.drawString("Paint Me!",100, 50);
```

The starting spot for the text is location (100, 50), 100 pixels from the left end and 50 pixels from the top. The font size is 40 and imitates Times New Roman (I think).

Finally, here is an example that draws a blue line slanting up towards the bottom portion of the frame:

```
g.setColor(Color.BLUE);
g.drawLine(50, 350, 450, 250);
```

In general, the best way to learn how to use the Graphics methods is to go to the graphics page (link provided in page 1 of the notes) and try them, one by one!