

# CSSE 220

## Intro to Java Graphics

Check out `IntroToJavaGraphics` from SVN. No quiz today.

# Announcement

- Exam 1 next week
- We're splitting the exam into written and programming and doing them on separate days
- Before next class
  - Complete the written portion of the 201510 written exam (provided on the schedule page)
  - Bring any questions you have to class
  - Be sure to time yourself to make sure you can complete it within the given 50 minutes

Simple Graphics

# **JAVA GRAPHICS**

# Simplest Java Graphics Program

```
import javax.swing.JFrame;
/**
 * From Ch 2, Big Java.
 * @author Cay Horstmann
 */
public class EmptyFrameViewer {
    /**
     * Draws a frame.
     * @param args ignored
     */
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.setSize(300,400);
        frame.setTitle("An Empty Frame");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

This code is already in your project for today

Creates a graphics frame object

Configures it

Display the frame

Tells Java to exit program when user closes the frame

**MyViewer** and **MyComponent** (Based on **RectangleViewer** and **RectangleComponent** from Big Java)

**LIVE CODING**

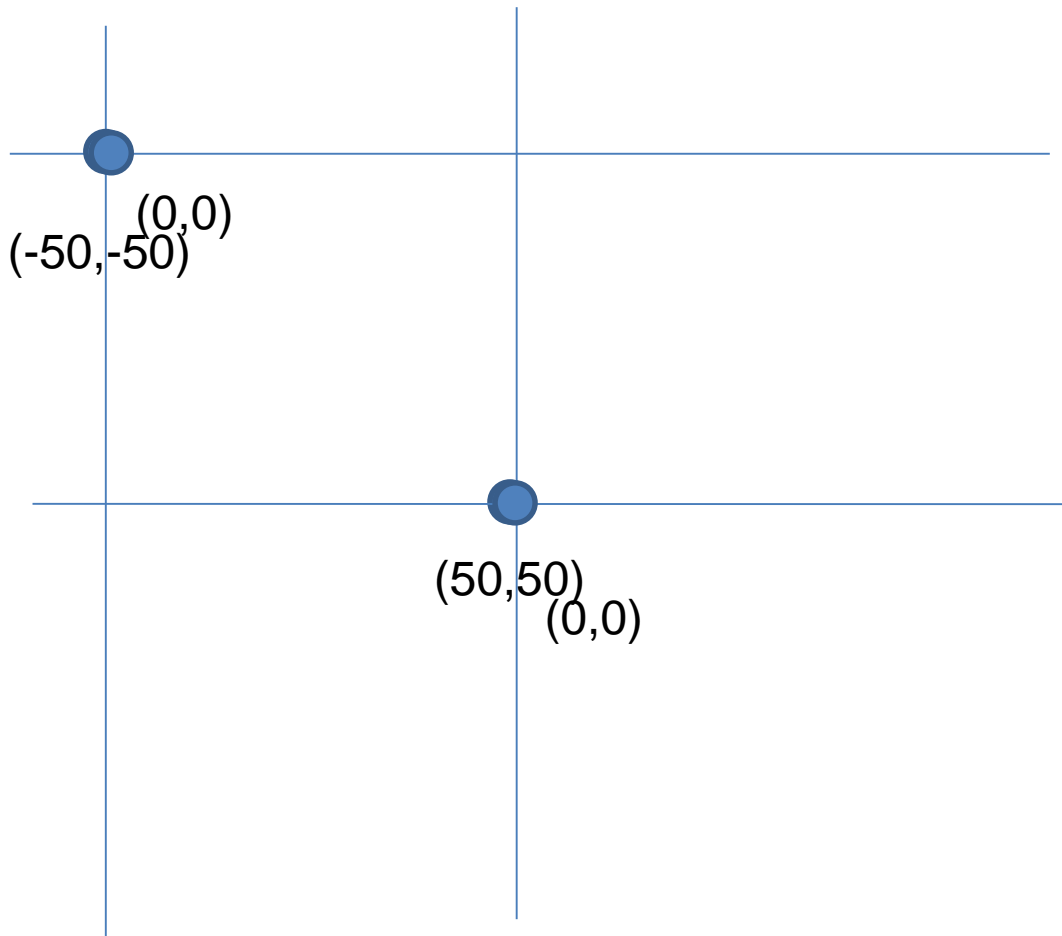
# Other Shapes

- `new Ellipse2D.Double(double x, double y,`  
• `double w, double h)`
- `new Line2D.Double(double x1, double y1,`  
• `double x2, double y2)`
- `new Point2D.Double(double x, double y)`
- `new Line2D.Double(Point2D p1, Point2D p2)`
- `new Arc2D.Double(double x, double y,`  
• `double w, double h,`  
• `double start, double extent,`  
• `int type)`
- `new Polygon(int[] x, int[] y, int nPoints);`
- **Try some of these!**
  - Add an ellipse and both kinds of lines to  
`MyComponent`

# Using translate and rotate successfully

- Translate and rotate to adjust the “state” of the pen
- It is usually easier to move the pen, then draw in a fixed configuration around (0,0), then move the pen back
- Make (0,0) your center of rotation
  - can change the point of origin using `translate()` so you can rotate different portions of the component

# Translate



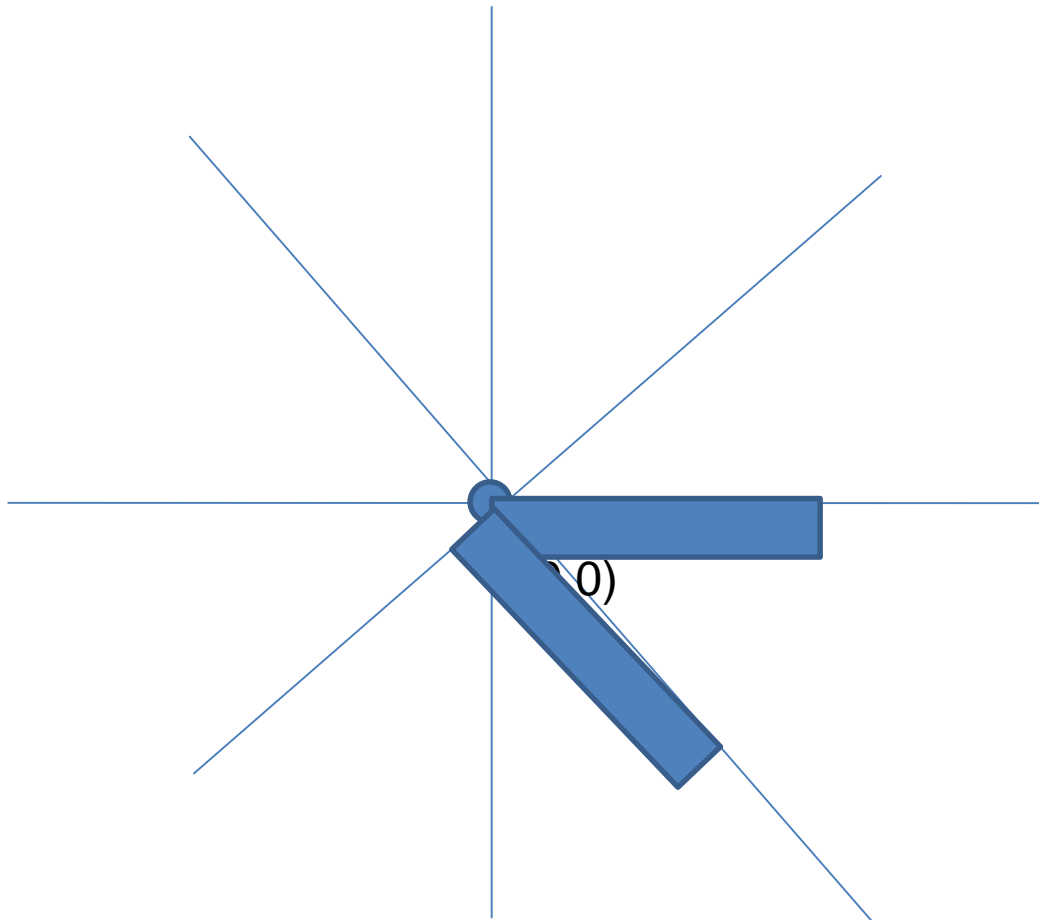
Originally, origin of 0,0  
at top left of screen (with (50,50)  
marked below)

If we called `g2.translate(50, 50)`,  
here's what would happen:

Always want to make sure we  
reset the pen, so when we're done,  
we need to translate back to where  
we started, in this case:  
`g2.translate(-50,-50)`



# Rotate



Let's say we've already translated to put the origin at (50,50) (mostly to make the slides look nicer)

If we drew a rectangle here like this:  
`g2.drawRect(0, 0, 50, 10);`, we would get something like...

What would happen if we called `g2.rotate(Math.PI/4);` (radians) then call `g2.drawRect(0, 0, 50, 10);` again?

Remember,  $y$  is positive down instead of up, so the rotate will go reverse of what you might be expecting

# Work

- Work on the 3 todos in the translationrotation package (TranslateComponent, RotateComponent)
- Then solve the HourTimer Problem
- Details are in the PDF within your repo

Scene project

# **SCENE INTRODUCTION**