CSSE 220

Intro to Java Graphics

Import IntroToJavaGraphics from the repo

Announcements

- Turn in Design HW #1
- Exam 1 this week
    - Two parts: written and programming

TeamGradeBook Debugging Technique

- Create `toString` for Team & Student classes
- At end of a “handle” operation, call `toString`
- For example at end of handleAddTeam add:
  `System.out.println(team.toString());`
- Set break point in JUnit test
  - right before call to “handle” operation
  - then “step over” call when break point is hit
  - look at toString’s output in Console window

Scene HW Assignment

1. Open Eclipse
2. From Project Explorer do a Team Pull
3. Then Import as usual

Go Over Design Problem #1

- Collect DP1
- Then go over DP1

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

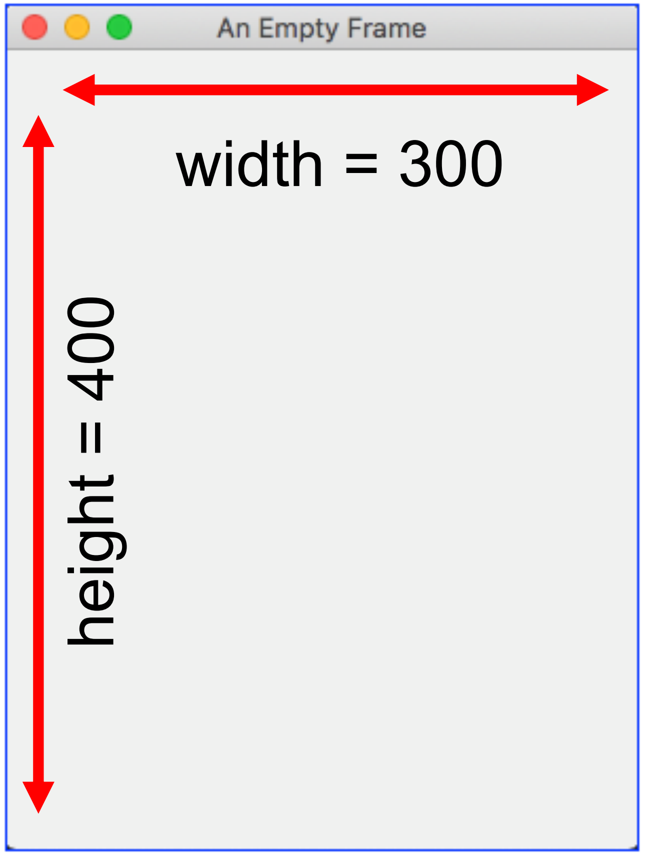
Creates a graphics frame object

Configures it

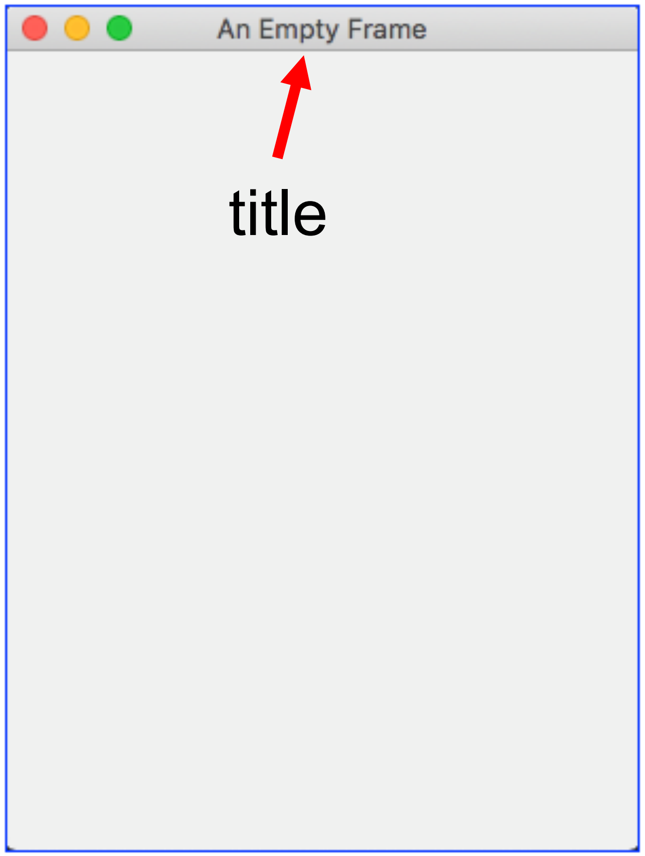
Tells Java to exit program when user closes the frame

Display the frame

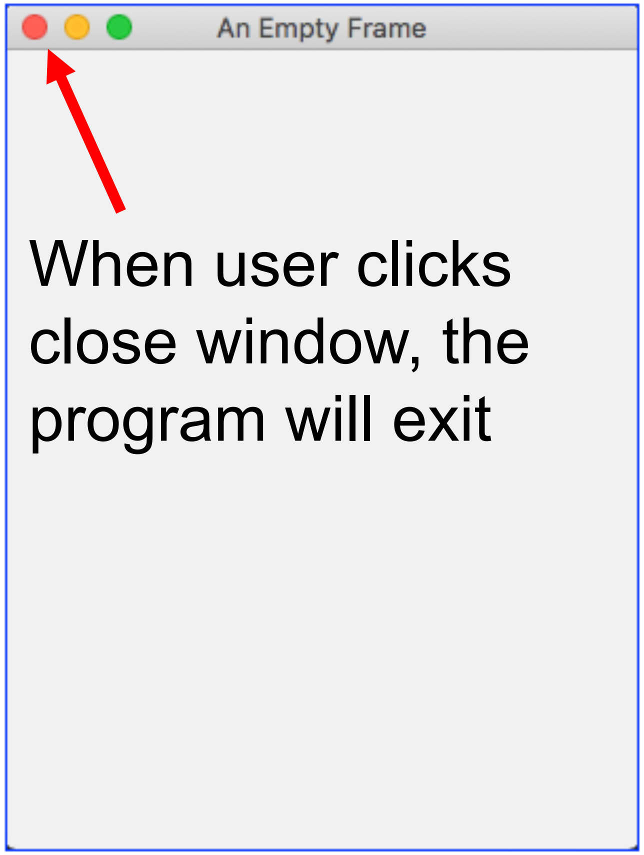
```
import javax.swing.JFrame;

public class EmptyFrameViewer {
  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);
  } // main
} // EmptyFrameViewer
```



```
import javax.swing.JFrame;

public class EmptyFrameViewer {
  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);
  } // main
} // EmptyFrameViewer
```

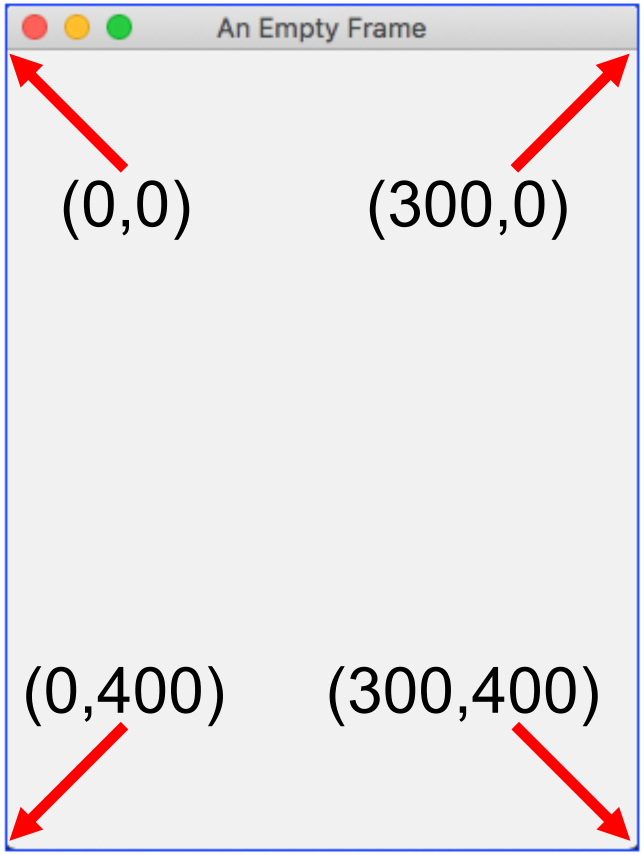


```
import javax.swing.JFrame;

public class EmptyFrameViewer {
   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);
   } // main
} // EmptyFrameViewer
```



When user clicks close window, the program will exit

```java
import javax.swing.JFrame;

public class EmptyFrameViewer {
   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);
   } // main
} // EmptyFrameViewer
```



(x, y) coordinate system

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

LIVE CODING

- EmptyFrameViewer from IntroToJavaGraphs pkg
- Add MyComponent
- Work on Graphics Activity – see link on CSSE220 Schedule, Day 7

- At bell between periods – should be able to answer first couple of quiz questions

Scene HW Assignment

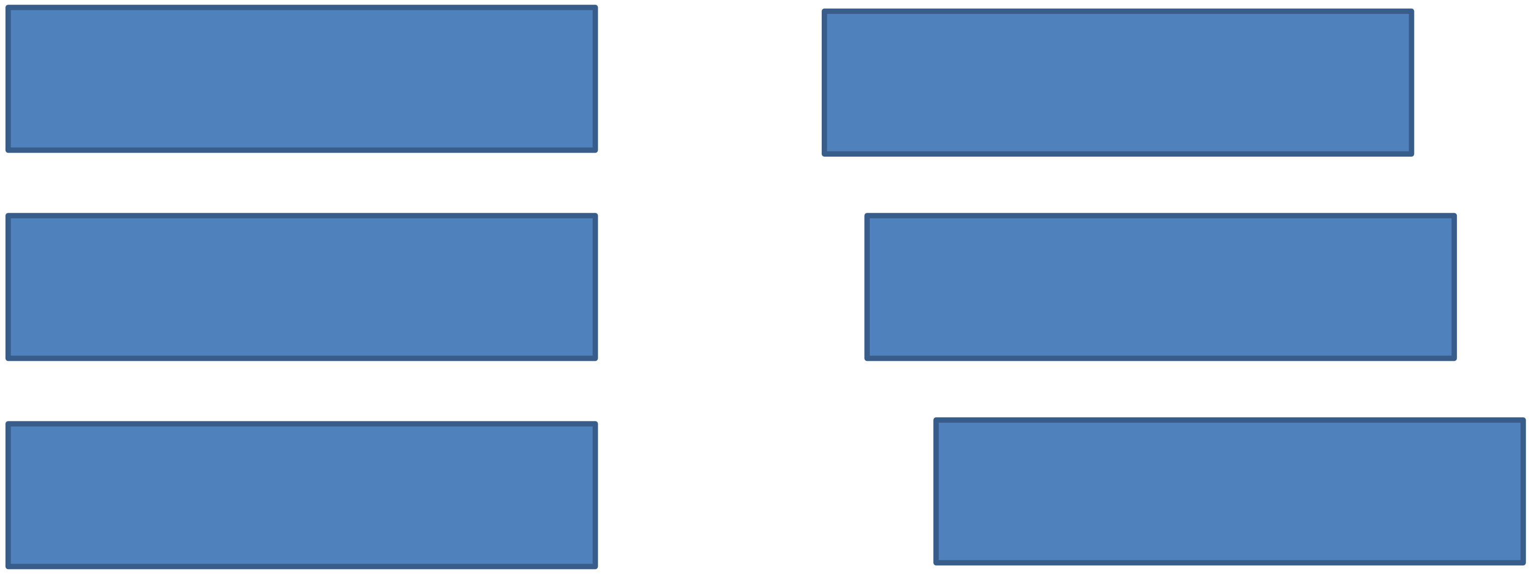
- Go to Moodle – look at Scene assignment requirements
- Now talk about translate and rotate

Other Shapes

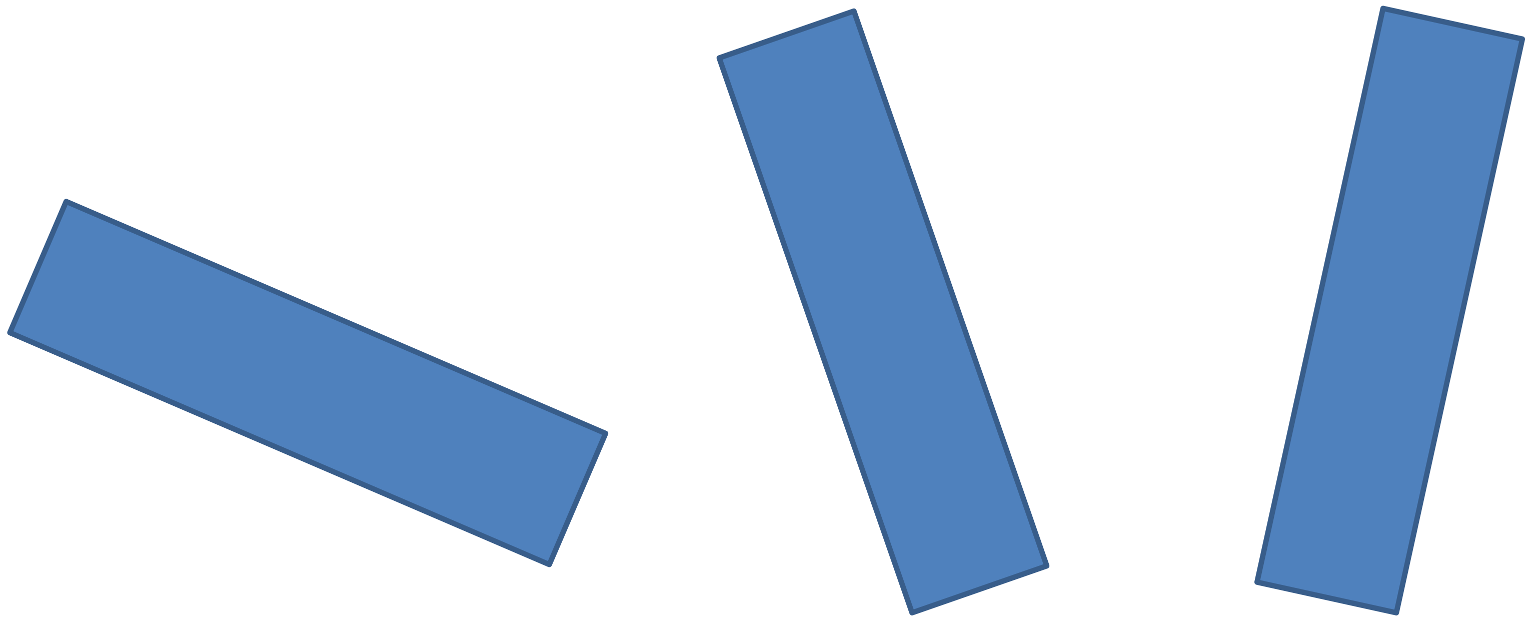
```
• new Ellipse2D.Double(double x, double y,
•                       double w, double h)
• new Line2D.Double(double x1, double y1,
•                    double x2, double y2)
• new Point2D.Double(double x, double y)
• new Line2D.Double(Point2D p1, Point2D p2)
• new Arc2D.Double(double x, double y,
             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

How to draw a shape at different positions?



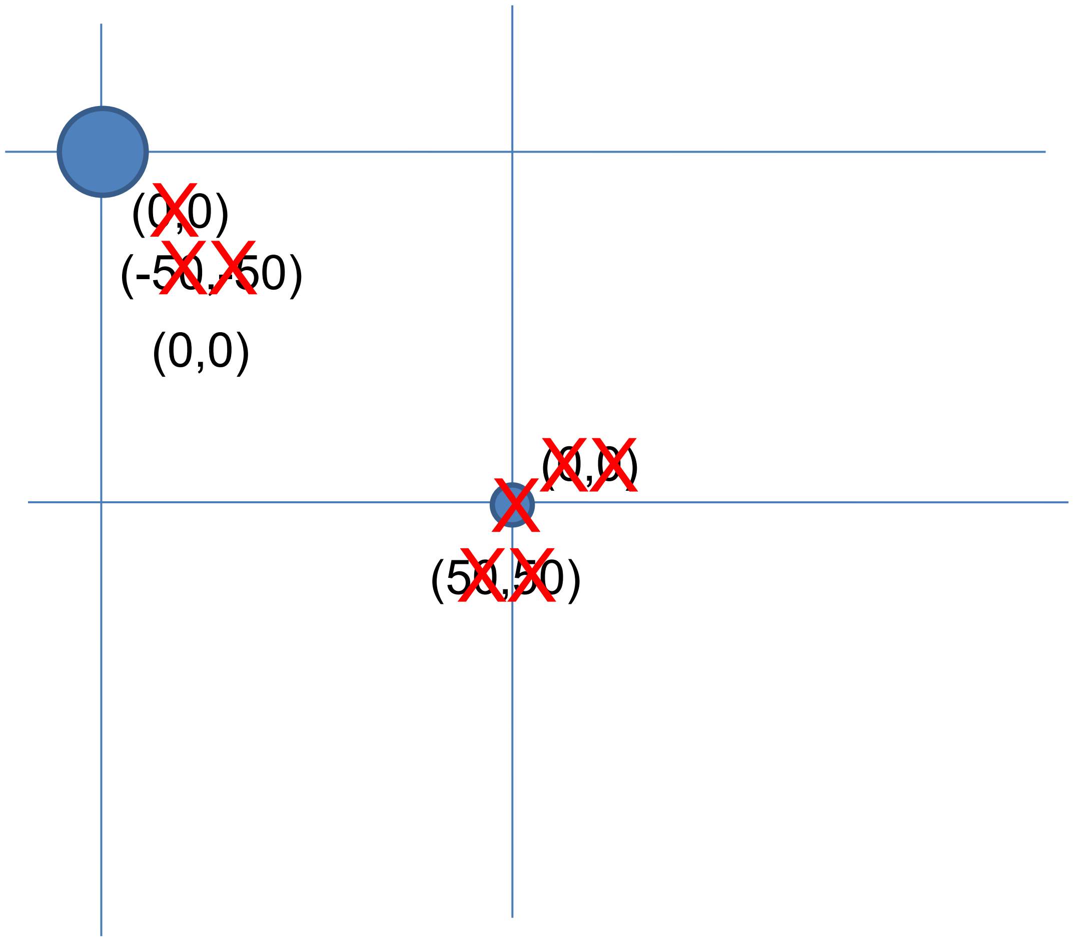
How to draw a rotated shape?



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

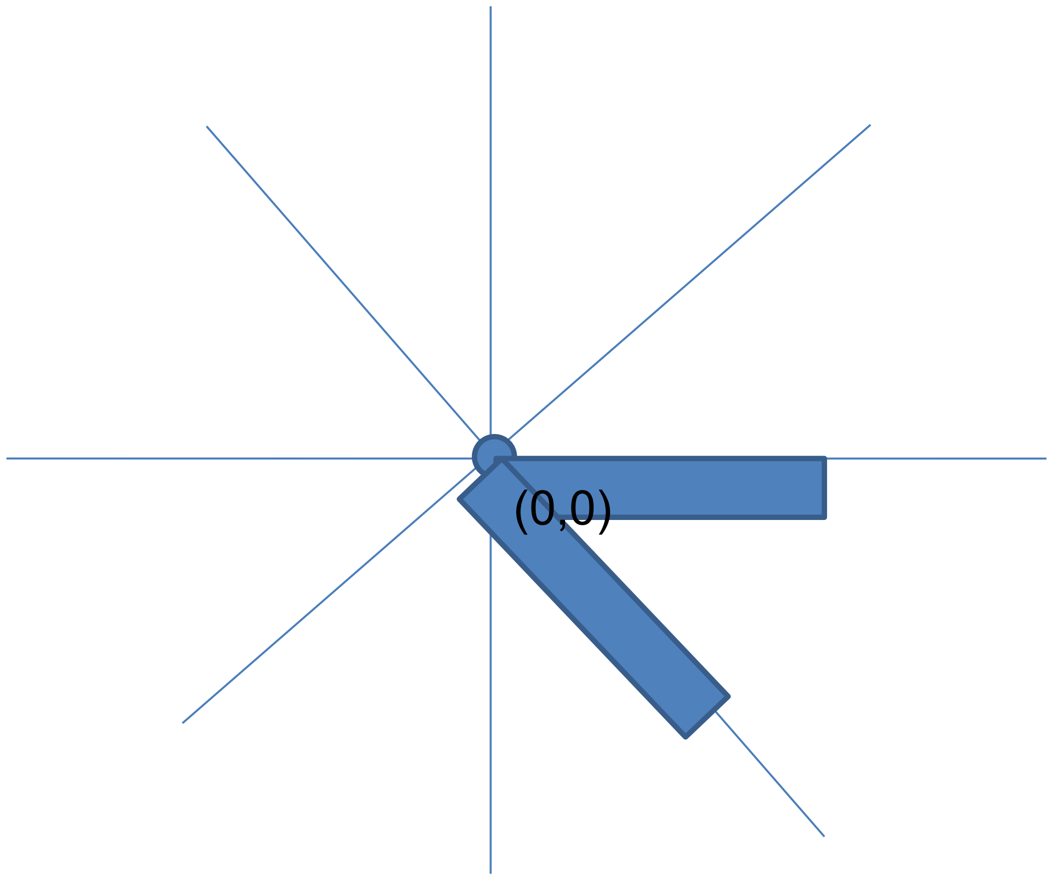


Originally, origin of 0,0
at top left of screen (with (50,50)
marked below)

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

Graphics Debugging

- Test each step as you go!
- First make sure you get something visible
- 1. translate
- 2. rotate
- 3. draw
- 4. un-rotate
- 5. un-translate

Scene project

SCENE INTRODUCTION