

CSSE 220

Intro to Java Graphics

Check out `IntroToJavaGraphics` and `BiggestFan` projects from SVN

Announcement

- Exam 1 Starts Tuesday the 20th
 - We're splitting the exam into written and programming and doing them on separate days
- Before Friday's class
 - Complete the written portion of the 201510 written exam (provided in class today or linked on the schedule page)
 - Bring any questions you have to Friday's class
 - Be sure to time yourself to make sure you can complete it within the given 50 minutes

Outline

- Static (by examples)
- Live coding: a Java graphics program

Understanding static

STATIC

Why fields can't always be static

```
public class Student {  
    private String name;  
    private char grade;  
  
    public Student(  
        String name,  
        char grade){  
        this.name = name;  
        this.grade = grade;  
    }  
}
```

```
@Override  
public String toString() {  
    return name +  
        " has a grade of "  
        + grade;  
    }  
}
```

```
public static void main(String[] args) {  
    Student a = new Student("Adam", 'A');  
    Student b = new Student("Bryan", 'B');  
    Student c = new Student("Chris", 'C');  
    System.out.println(a);  
    System.out.println(b);  
    System.out.println(c);  
}
```

OUTPUT:

```
Adam has a grade of A  
Bryan has a grade of B  
Chris has a grade of C
```

Why not make the grade static?

```
public class Student {  
    private String name;  
    private static char grade;  
  
    public Student(  
        String name,  
        char grade){  
        this.name = name;  
        this.grade = grade;  
    }  
}
```

```
@Override  
public String toString() {  
    return name +  
        " has a grade of "  
        + grade;  
}  
}
```

```
public static void main(String[] args) {  
    Student a = new Student("Adam", 'A');  
    Student b = new Student("Bryan", 'B');  
    Student c = new Student("Chris", 'C');  
    System.out.println(a);  
    System.out.println(b);  
    System.out.println(c);  
}
```

OUTPUT:

```
Adam has a grade of C  
Bryan has a grade of C  
Chris has a grade of C
```

Static means there's only one instance of a field/method for every instance of a class that's created. So when you change a grade, they all change.

When do we make methods static?

- Utility Methods
 - Things like `abs`, `sqrt`, etc.
 - Don't need an instance of a class to run them
- How do I know?
 - No references to non-static fields/methods
 - No "this" keyword used in method

```
public class Car {

    double mileage;

    //other stuff

    public double getMilesTravelled() {
        return this.mileage;
    }

    public static double convertMilesToKm(double numberOfMiles) {
        return numberOfMiles * 1.609344f;
    }

}

//Elsewhere...

//requires you to have a car object
Car myCar = new Car();
//requires you to have a car object
System.out.println(myCar.getMilesTravelled());//output depends on code
//can be called on the class Car itself
System.out.println(Car.convertMilesToKm(77));//output is 123.919488
```

```
public class Bicycle {
```

```
    private int speed;
```

```
    private static int numCreated = 0;
```

```
    public Bicycle(int speed) {
```

```
        this.speed = speed;
```

```
        Bicycle.numCreated++;
```

```
    }
```

```
    public int getSpeed() {
```

```
        return this.speed;
```

```
    }
```

```
    public static int getNumCreated() {
```

```
        return Bicycle.numCreated;
```

```
    }
```

```
}
```

```
//No requirement to have a Bicycle yet..
```

```
System.out.println(Bicycle.getNumCreated());
```

```
Bicycle myBike1 = new Bicycle(18);
```

```
Bicycle myBike2 = new Bicycle(1);
```

```
System.out.println(Bicycle.getNumCreated() + " " + myBike1.getSpeed());
```

```
0
```

```
2 18
```

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

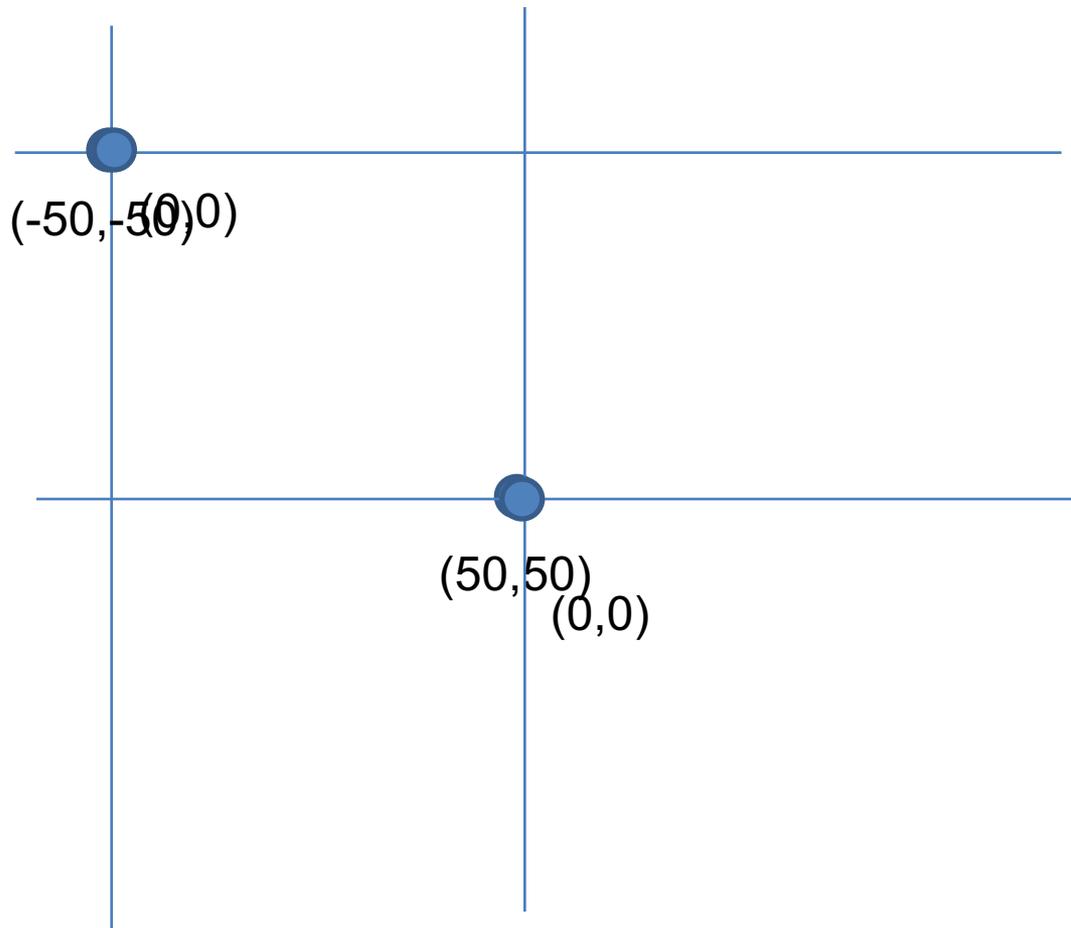
Other Shapes

- `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

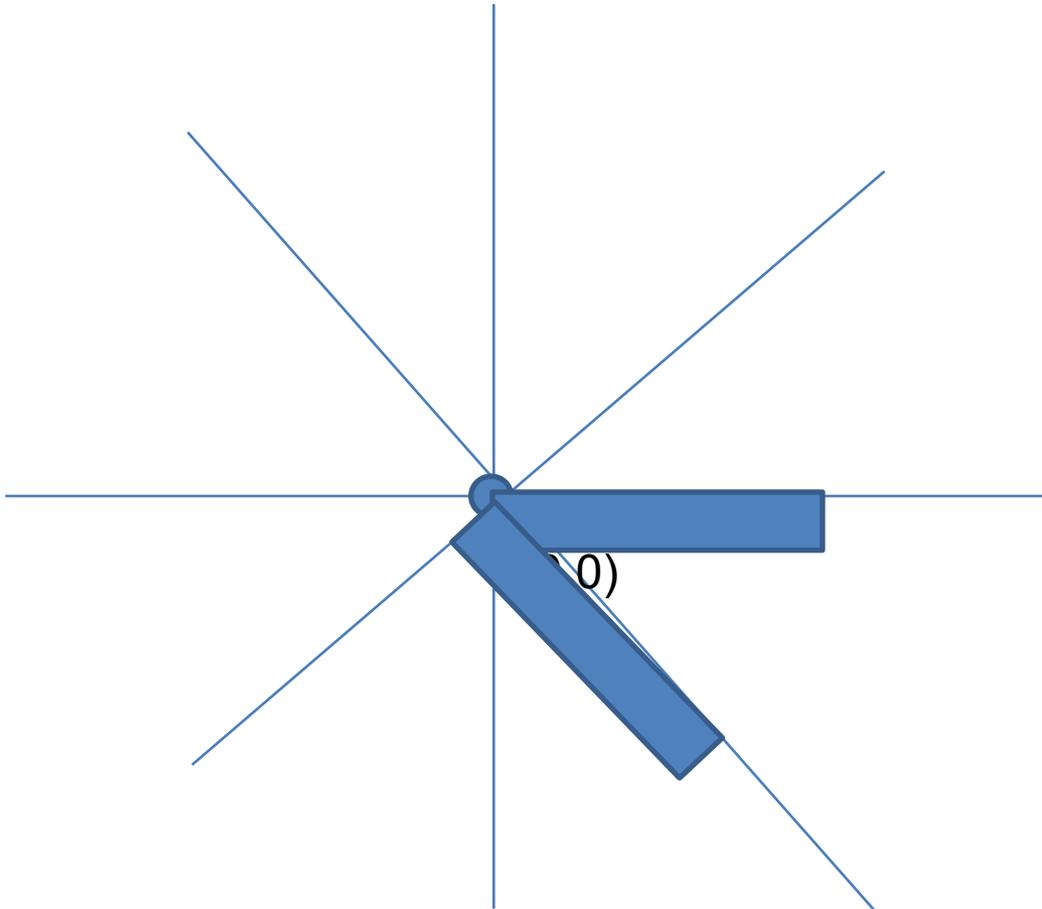


Originally, origin of 0,0
at top left of screen (with (50,50)
marked below)

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 on the biggest fan code

- We'll walk through it together to explain how the classes work
- Then you should modify the fan to print one blade vertically – use transform to move (0,0) to the **center** of the fan and then draw from there

Scene project

SCENE INTRODUCTION