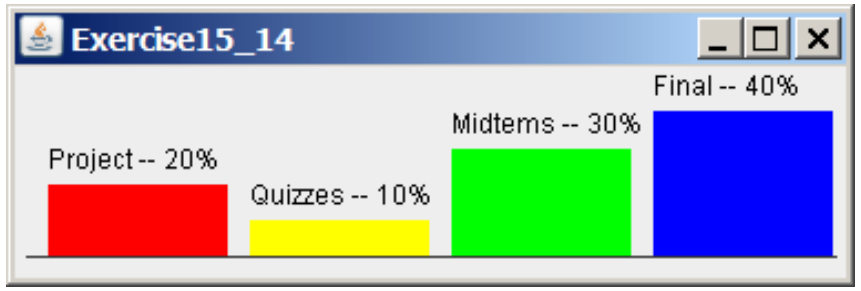
Java GUI Programming
AWT/SWING - Graphics

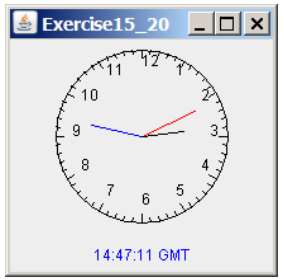
OVERVIEW OF GRAPHICS (JPANEL+GRAPHICS)

ERIC Y. CHOU, PH.D. IEEE SENIOR MEMBER

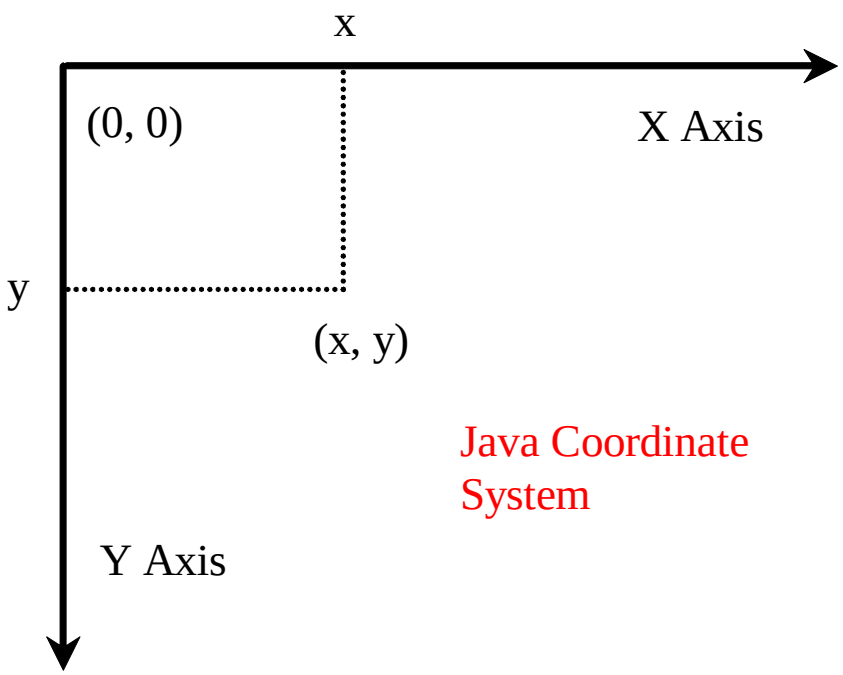
Graphical Representation

If you want to draw shapes such as a bar chart, a clock, or a stop sign, how do you do it?

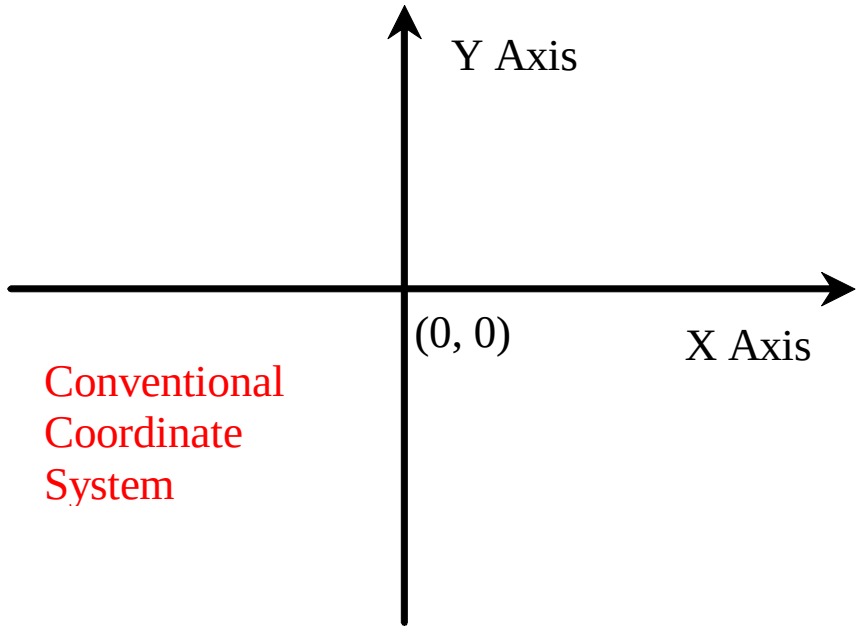




Java Coordinate System

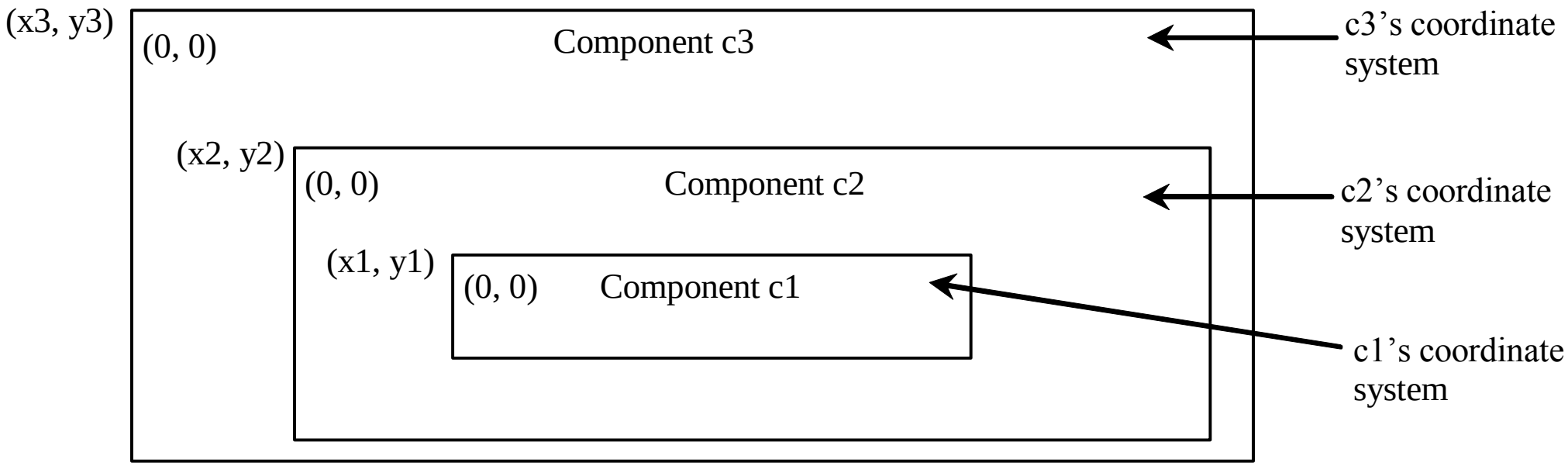


Java Coordinate System



Conventional Coordinate System

Each GUI Component Has its Own Coordinate System



The Graphics Class

You can draw strings, lines, rectangles, ovals, arcs, polygons, and polylines, using the methods in the Graphics class.

java.awt.Graphics	
+setColor(color: Color): void	Sets a new color for subsequent drawings.
+setFont(font: Font): void	Sets a new font for subsequent drwings.
+drawString(s: String, x: int, y: int): void	Draws a string starting at point (x, y).
+drawLine(x1: int, y1: int, x2: int, y2: int): void	Draws a line from (x1, y1) to (x2, y2).
+drawRect(x: int, y: int, w: int, h: int): void	Draws a rectangle with specified upper-left corner point at (x, y) and width w and height h.
+fillRect(x: int, y: int, w: int, h: int): void	Draws a filled rectangle with specified upper-left corner point at (x, y) and width w and height h.
+drawRoundRect(x: int, y: int, w: int, h: int, aw: int, ah: int): void	Draws a round-cornered rectangle with specified arc width aw and arc height ah.
+fillRoundRect(x: int, y: int, w: int, h: int, aw: int, ah: int): void	Draws a filled round-cornered rectangle with specified arc width aw and arc height ah.
+draw3DRect(x: int, y: int, w: int, h: int, raised: boolean): void	Draws a 3-D rectangle raised above the surface or sunk into the surface.
+fill3DRect(x: int, y: int, w: int, h: int, raised: boolean): void	Draws a filled 3-D rectangle raised above the surface or sunk into the surface.
+drawOval(x: int, y: int, w: int, h: int): void	Draws an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+fillOval(x: int, y: int, w: int, h: int): void	Draws a filled oval bounded by the rectangle specified by the parameters x, y, w, and h.
+drawArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void	Draws an arc conceived as part of an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+fillArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void	Draws a filled arc conceived as part of an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+drawPolygon(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a closed polygon defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.
+fillPolygon(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a filled polygon defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.
+drawPolygon(g: Polygon): void	Draws a closed polygon defined by a Polygon object.
+fillPolygon(g: Polygon): void	Draws a filled polygon defined by a Polygon object.
+drawPolyline(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a polyline defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.

Basic Java Graphics

The simplest to draw graphics in Java is to extend **JPanel**, a Swing component, and override its **paintComponent (Graphics g)** method in order to draw on the graphics object **g**. Whenever Java tries to render a Swing GUI component, it calls the component's paintComponent (Graphics g) method with the current graphics context as the parameter. In the code for paintComponent (Graphics g), you almost always call **super.paintComponent (g)** in order to get the correct internal (hidden) rendering sequence. The code for BodyPartsCanvas illustrates this process.

BodyPartsCanvas

A sub-class of JPanel as Graphical Component Holder


```
public class BodyPartsCanvas extends JPanel
{
   // Other fields and methods...
   public void paintComponent (Graphics g)
   {
      super.paintComponent (g);
      // code to draw on g....
   }
}
```

You never call paintComponent (Graphics g) directly.

Instead, you should call **repaint()** to let Java schedule the repaint process and properly call paintComponent.

paintComponent Example

Demo Program: TestPaintComponent.java

In order to draw things on a component, you need to define a class that extends JPanel and overrides its paintComponent method to specify what to draw. The first program in this chapter can be rewritten using paintComponent.

Drawing Graphics on Panels

- `JPanel` can be used for both containing components and for direct drawing
- To draw in a `JPanel`, you create a new class that extends `JPanel` and override the `paintComponent` method
- Doing this prevents you from interfering with other components
- Override this method

```
protected void paintComponent(Graphics g)
```

- g is provided automatically by JVM