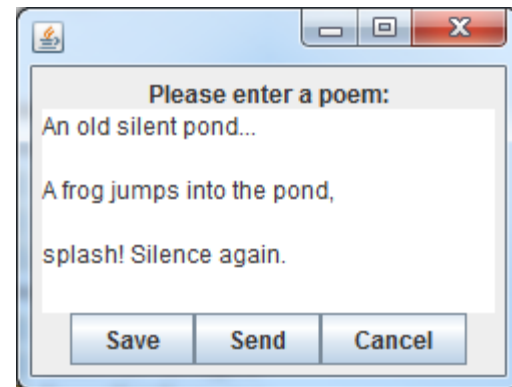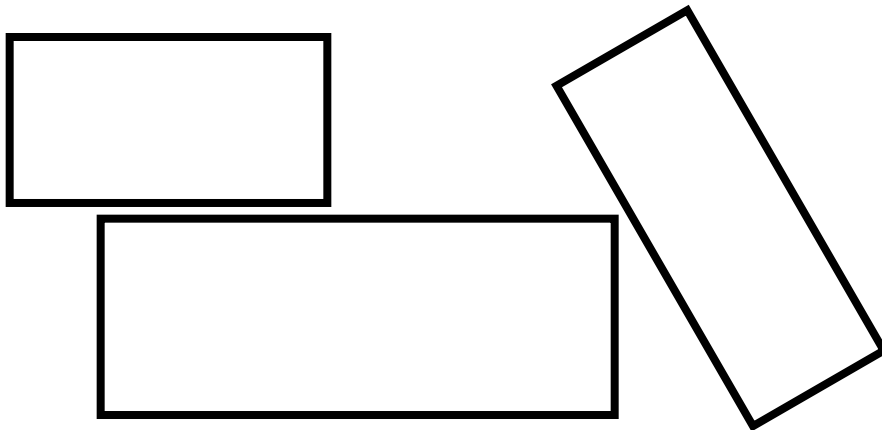# COMPSCI 230

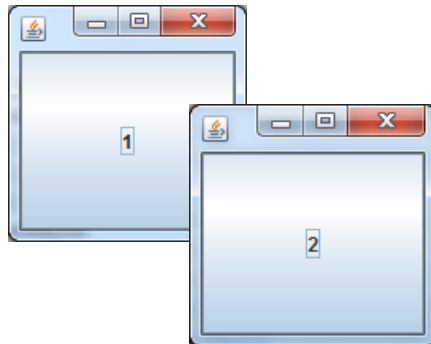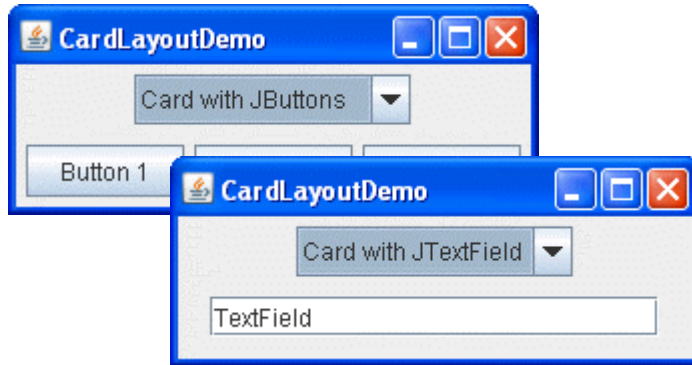## Software Design and Construction

Drawing

2013-05-08

Layout Managers

CardLayout

- Layout that contains different widgets at different times
- Used for switching between different UI screens in the same window / panel
- Like changing the card on the top of a stack of cards

```
...
CardLayout cardLayout =
    new CardLayout();
setLayout(cardLayout);

add(new JButton("1"), "Screen 1");
add(new JButton("2"), "Screen 2");

cardLayout.show(getContentPane(),
    "Screen 2");
...
```

http://docs.oracle.com/javase/tutorial/uiswing/layout/flow.html

GridBagLayout

- Widgets arranged in table (aka. a grid) with rows, columns and cells
- Position and size properties of each widget are specified in a `GridBagConstraints` object
  - `gridx` for start column index
  - `gridy` for start row index
  - `gridwidth` for column span (i.e. width in columns)
  - `gridheight` for row span (i.e. height in rows)
  - `weightx` and `weighty` for relative size
  - `ipadx, ipady, insets` for margins around widgets
  - `fill` to set whether to fill extra space with the widget
- One of the most popular, also in other toolkits (e.g. HTML tables)

http://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html
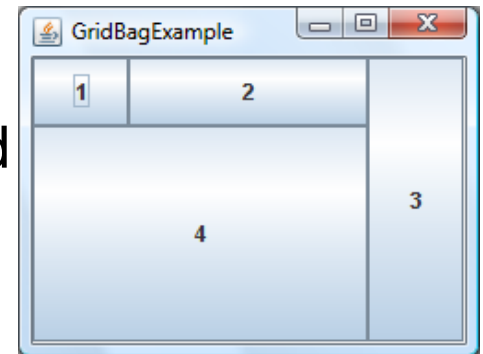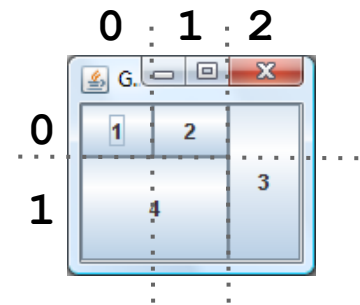
GridBagLayout Example

```
panel.setLayout(
    new GridBagLayout());
GridBagConstraints c =
    new GridBagConstraints();

JButton b1 = new JButton
    ("1");
c.gridx = 0; c.gridy = 0;
c.weightx = 0.1;
c.weighty = 0.1;
c.fill =
    GridBagConstraints.BOTH;
panel.add(b1, c);




JButton b2 = new JButton
    ("2");
c.gridx = 1; c.gridy = 0;
c.weightx = 0.8;
panel.add(b2, c);
```

```
JButton b3 = new JButton
    ("3");
c.gridx = 2; c.gridy = 0;
c.gridheight = 2;
c.weightx = 0.1;
panel.add(b3, c);

JButton b4 = new JButton
    ("4");
c.gridx = 0; c.gridy = 1;
c.gridheight = 1;
c.gridwidth = 2;
c.weighty = 0.8;
panel.add(b4, c);
```
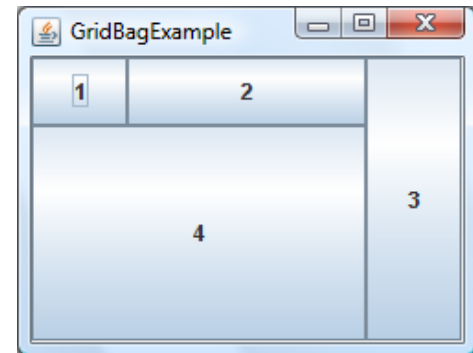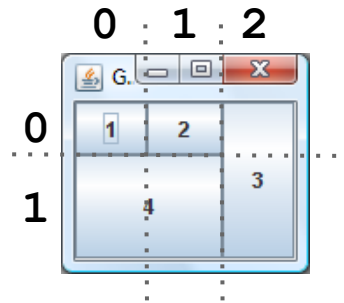
Constraint-Based Layout

- Modern way of specifying layout, e.g. Apple AutoLayout
- Grid lines are variables with coordinates (*tabs*)
- Place controls by choosing left, top, right and bottom tab:

$$a =_{def} (x_1, y_1, x_2, y_2, layer, content)$$

- Positions and sizes are specified with linear constraints, e.g.

$$x_2 - x_1 = 2(x_4 - x_3)$$

- Layout is calculated with a constraint solver

Example:

$$A = \{(x_0, y_0, x_2, y_1, 0, red), (x_2, y_0, x_3, y_2, 0, green),$$
$$(x_1, y_2, x_3, y_3, 0, blue), (x_0, y_1, x_1, y_3, 0, grey),$$
$$(x_1, y_1, x_2, y_2, 0, empty)\}$$
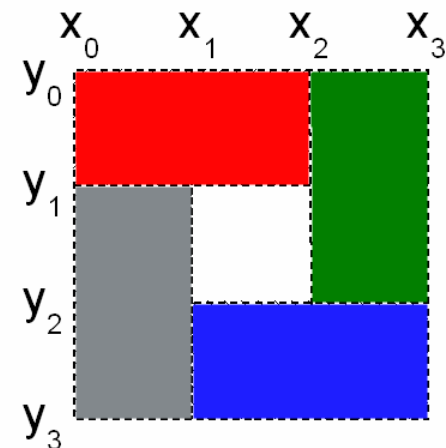
Layout Managers and WindowBuilder

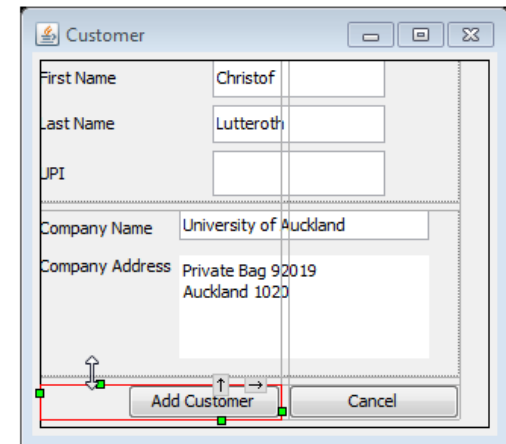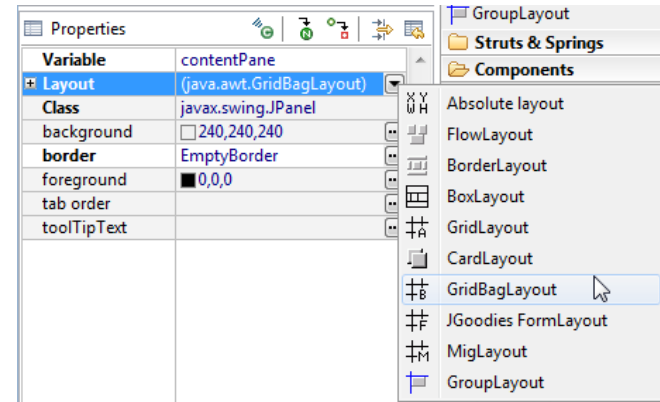WindowBuilder supports the common Swing layout managers

- Absolute layout = no layout manager i.e. only fixed positions and sizes
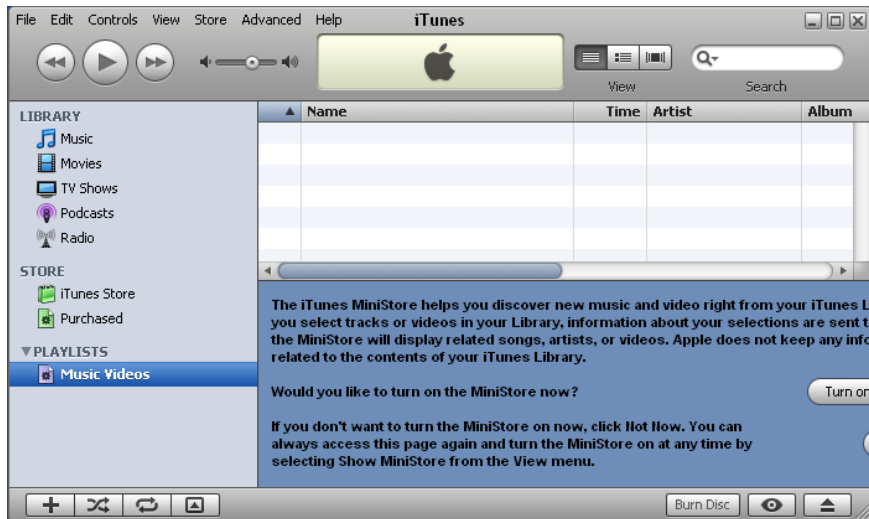
Layouts can be edited visually, to a degree

- Drag & drop

- Manual coding necessary sometimes

Support for converting layouts, e.g.

- Start with absolute layout (easy)

- Then just switch to GridBagLayout to make UI somewhat resizable

Custom Widgets
and Drawing

Recap: Rendering of Widgets

Widgets have a visual representation

- Widgets define "paint" event listener: draws the widget by sending commands to the windowing system
- Widget gets paint events (aka. "update events") from the windowing system (through GUI framework)
- Often not complete redrawing, but "update region"
- Application can send "invalidate" events to the windowing system if redrawing necessary (to trigger paint events)
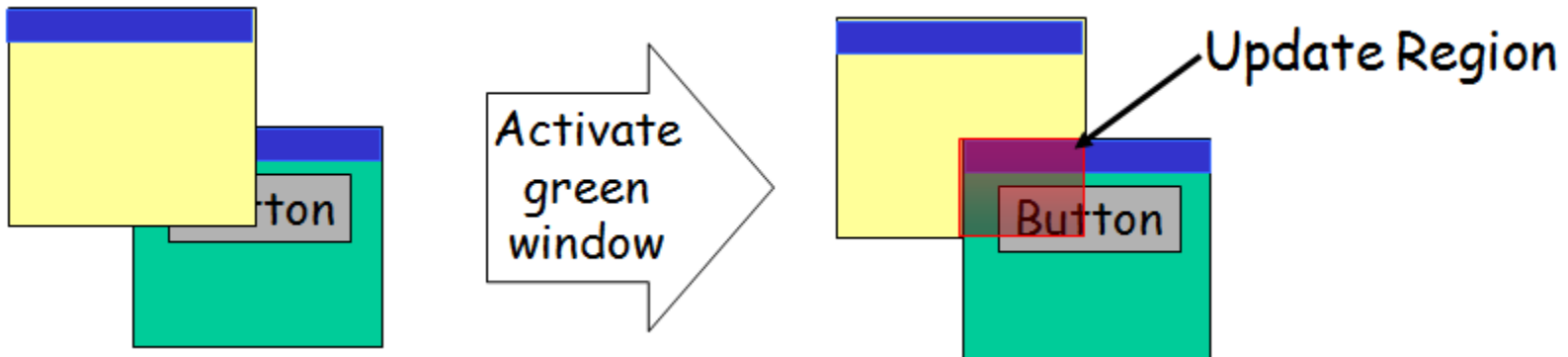
How Swing Widgets are Painted

All Swing widgets inherit from **JComponent**
- **JComponent** defines **paint(Graphics g)**
- **paint()** called by the system whenever drawing is necessary

paint() calls other methods of **JComponent**
- **paintComponent()** paints the widget (override this!)
- **paintBorder()** paints a border the widget may have (see **border** property)
- **paintChildren()** paints the children of the widget, if it is a container (don't override this!)

```
Button
```
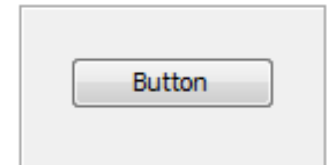
You never call **paint()** directly
- Instead invalidate the widget region by calling **repaint()**
- **repaint()** asynchronously calls **paint()** (through windowing system)
- You can give dirty region as argument: **repaint(Rectangle r)**

Creating a Custom Widget

1. Create new class that extends **JPanel**
2. Override **paintComponent(Graphics g)** with custom drawing code
   - Make sure to honor the **width** and **height** of the widget
   - Possibly call **super.paint(g)** to draw the superclass widget (e.g. unicolored background)
3. Override **getPreferredSize()** to return the right preferred size for your widget

```
class MyPanel extends JPanel {
   public void paintComponent(Graphics g) {
      super.paintComponent(g);
      g.drawString("Hello World!",10,20);
   }

   public Dimension getPreferredSize() {
      return new Dimension(100,50);
   }  }
```
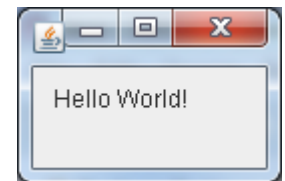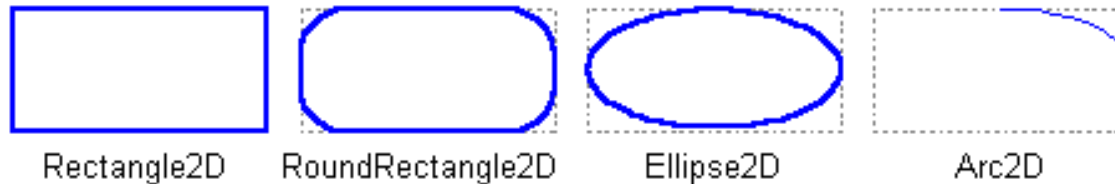
Drawing with Java

Always draw in a graphics context (`Graphics / Graphics2D`):

- Uniform way to draw on different devices, like a universal canvas
- Properties of the current pen used for drawing
  - `color`, `background`, `font`
  - `stroke`, i.e. pen size and shape
  - `paint`, i.e. a color pattern to use
  - `composite` type, i.e. how it looks when shapes are drawn onto existing shapes (e.g. blending them together)
  - `clip`ping rectangle to limit painting area
- Drawing methods, e.g. `draw(Shape), fill(Shape), drawString(), drawImage()`
- Transformation methods to apply to the drawing operations, e.g. `scale(), rotate(), translate()`

Drawing Basics

(0, 0)

- By default, coordinate space origin in the top-left corner

- Shapes and lines represented as classes implementing interface **Shape**

Rectangle2D   RoundRectangle2D   Ellipse2D   Arc2D

Draw ⟶
Fill ⟶

Draw String ⟶ Text rendering
Text rendering
Text rendering
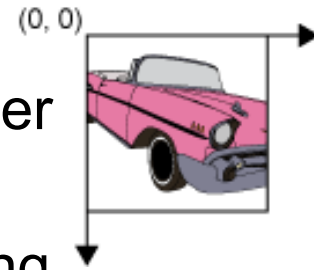
Draw Image ⟶

- ○ **draw(Shape s)** draws shape outline using **color** and **stroke**
- ○ **fill(Shape s)** draws solid shape using **color** / **paint**
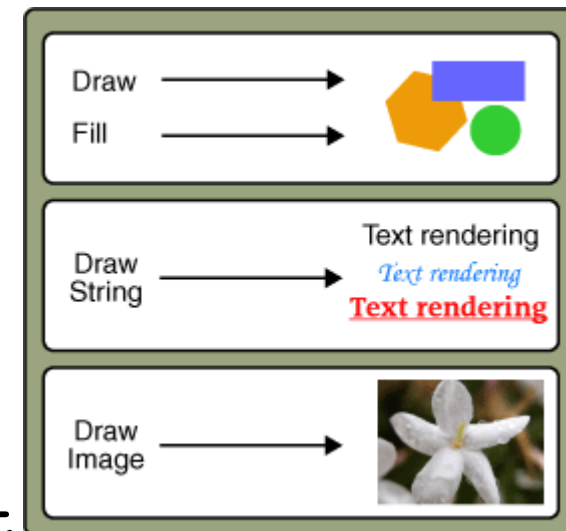
- **drawString(String s, float x, float y)** draws text using **font**
- **drawImage(Image i, int x, int y, ...)**

RoundedButton Part 1

```java
class RoundedButton extends JButton {
  public void paintComponent(Graphics g) {
    // Argument of paint() is actually a Graphics2D object,
    // which has more functionality than Graphics
    Graphics2D g2 = (Graphics2D) g;

    // Switch on anti-aliasing, which looks better
    g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
            RenderingHints.VALUE_ANTIALIAS_ON);
    g2.setRenderingHint(RenderingHints.KEY_TEXT_ANTIALIASING,
            RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
```
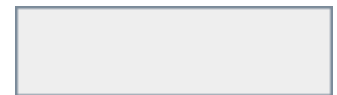
Without Anti-Aliasing: With Anti-Aliasing:

```java
    g2.setColor(getBackground());
    g2.fill(new Rectangle2D.Float(
        0, 0, getWidth(), getHeight()));

    g2.setColor(new Color(110, 120, 210));
    g2.fill(new RoundRectangle2D.Float(
        0, 0, getWidth(), getHeight(), 50, 50));

    ...
```

RoundedButton Part 2

```
    g2.setColor(new Color(120, 130, 255));
    g2.setStroke(new BasicStroke(5));
    g2.draw(new RoundRectangle2D.Float(
        2, 2, getWidth() - 4, getHeight() - 4, 50, 50));
    g2.setStroke(new BasicStroke(1));

    FontMetrics metrics = g2.getFontMetrics(getFont());
    int h = metrics.getAscent();
    int w = metrics.stringWidth(getText());

    g2.setColor(getForeground());
    g2.drawString(getText(),
        (getWidth() - w) / 2, (getHeight() + h) / 2);
  }

  public static void main(String[] args) {
    JFrame frame = new JFrame();
    RoundedButton r = new RoundedButton(); r.setText
("Hello!");
    r.setFont(new Font("Comic Sans MS", Font.PLAIN, 16));
    frame.getContentPane().add(r);
    frame.pack(); frame.setVisible(true);
} }
```

Summary

- **Drawing** can be performed using graphics objects
  - A graphics context (`Graphics2D`)
  - `Stroke`s, `Font`s, `Color`s...
  - `Shape` objects that can be `draw`n or `fill`ed
- **Custom components** can be created by overriding the method paintComponent(Graphics g) of a widget

References:
- The Java Tutorials: 2D Graphics.
  http://docs.oracle.com/javase/tutorial/2d/
- The Java Tutorials: Performing custom painting. http://docs.oracle.com/javase/tutorial/uiswing/painting/

Quiz

1. What does the **repaint()** method of a Swing widget do?
2. What does the **paintComponent()** of a Swing widget method typically do?
3. What is a graphics context?