Java graphics & event-driven UIs

- Java event-delegation model
- Swing highlights
- ⇒ Swing components survey
- Java graphics

JFrames

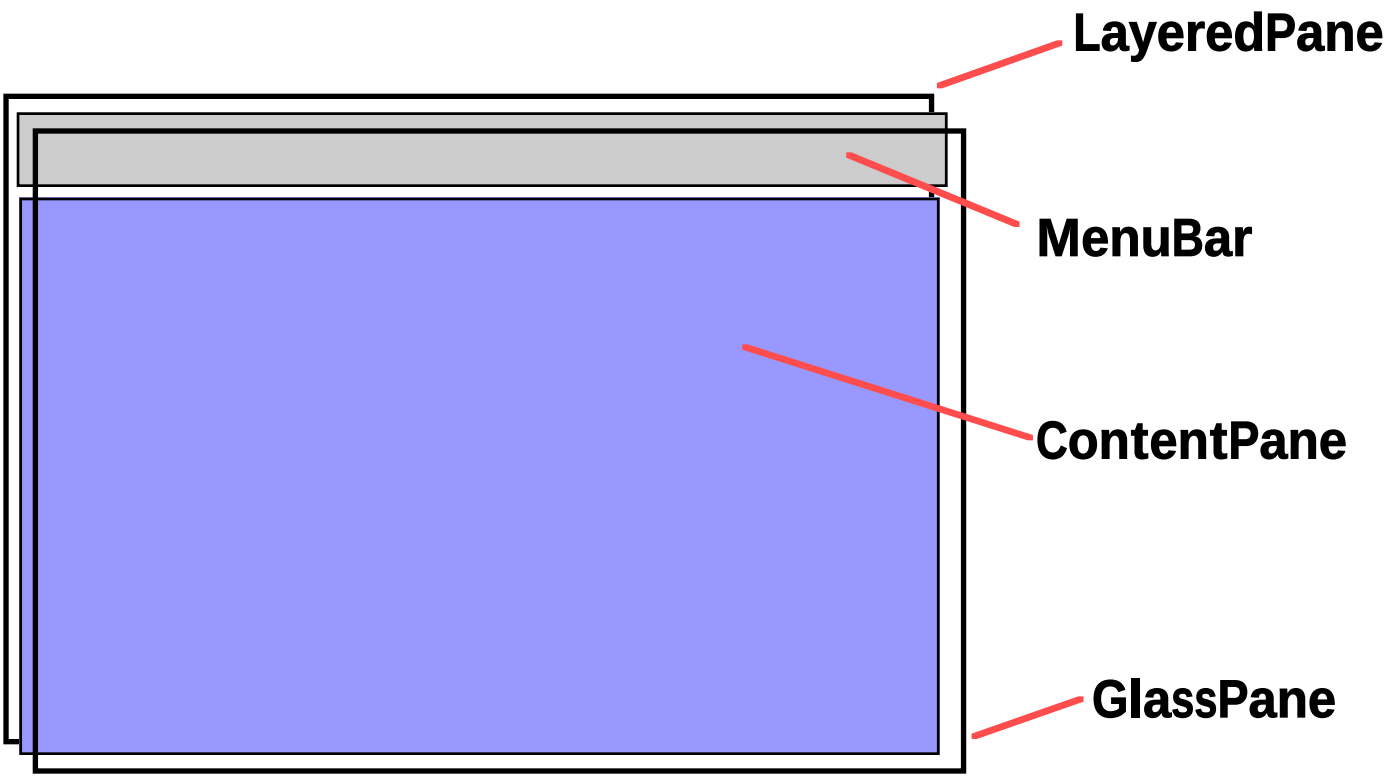
JFrames have layered ‘panes’
allows for things like
- floating pallettes
- overlapping internal windows
- menus attached to frames

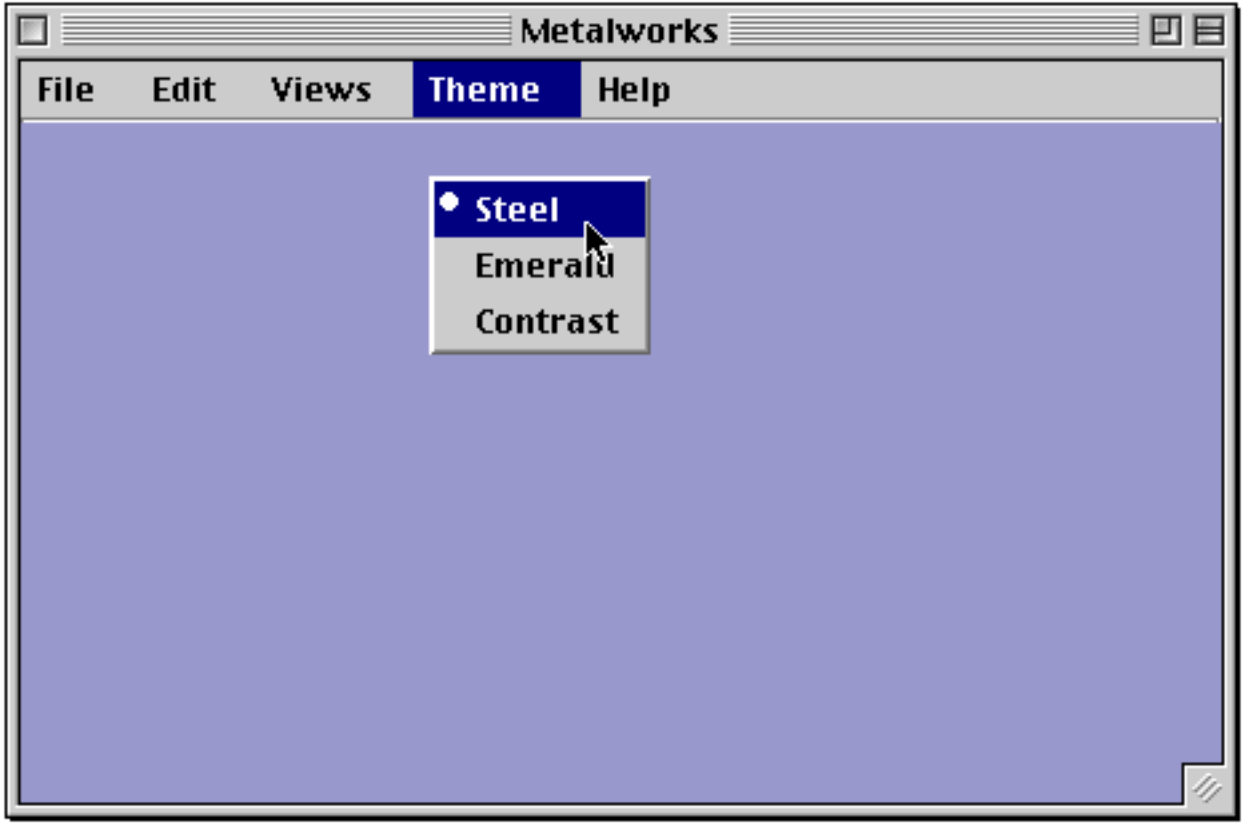
`JFrame` panes

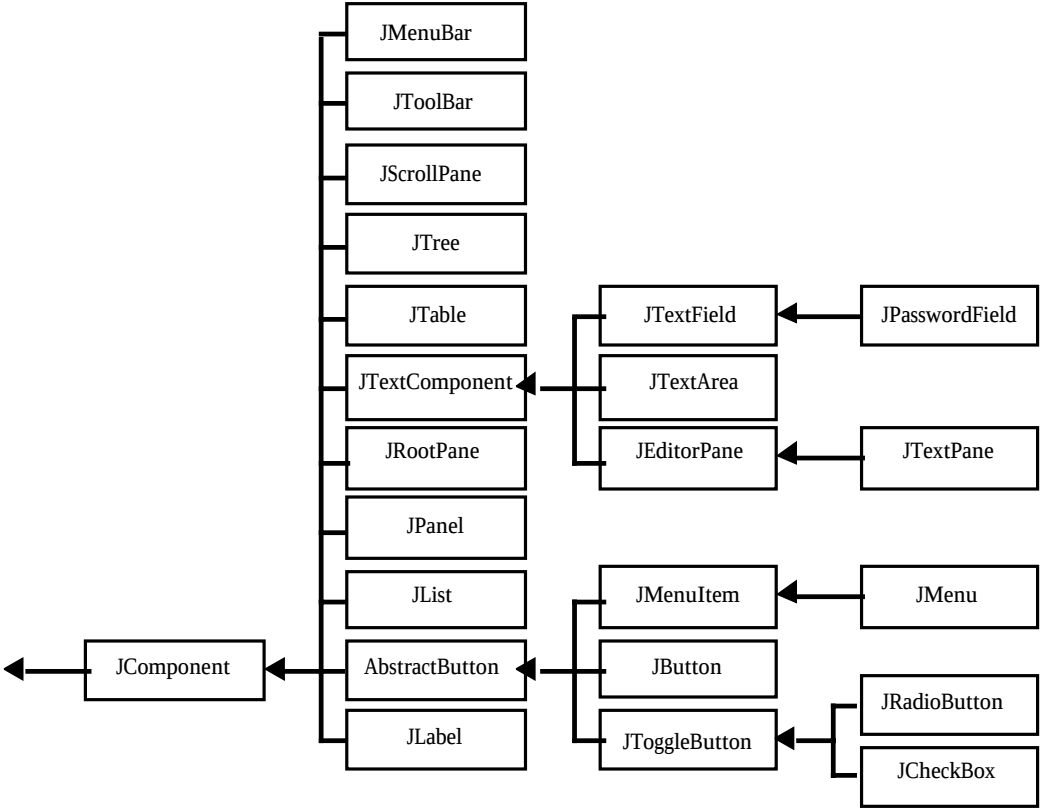
main child container is a `JRootPane`

this contains

- **GlassPane – where GUI gestures can take place**
- **LayeredPane – supports overlapping layers**
  - ***among others, contains***
    - **ContentPane – your components go here**
    - **MenuBar**


LayeredPane
MenuBar
ContentPane
GlassPane


Metalworks
File
Edit
Views
Theme
Help
Steel
Emerald
Contrast


JMenuBar
JToolBar
JScrollPane
JTree
JTable
JTextComponent
JRootPane
JPanel
JList
JComponent
AbstractButton
JLabel
JTextField
JPasswordField
JTextArea
JEditorPane
JTextPane
JMenuItem
JMenu
JButton
JRadioButton
JToggleButton
JCheckBox

Swing components

```
JFrame
JPanel
JLabel
JButton
JRadioButton
JCheckBox
```

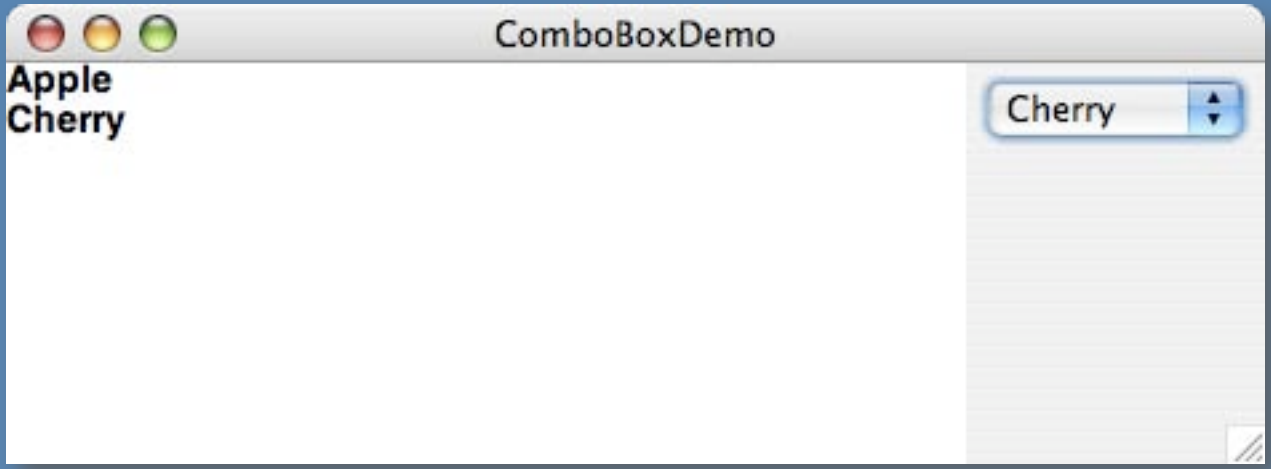
```
JToolTip
JScrollPane
JOptionPane
JTabbedPane
JTextField
JEditorPane
```

Swing components

Others:

- `JComboBox`
- `JToolBar`
- `JMenu`
- `JList`
- `JSplitPane`
- `JInternalFrame`
- `JTable`
- `JTree`
- `JSlider`
- `JProgressBar`
- `JTextPane`
- `JTextArea`

ComboBox

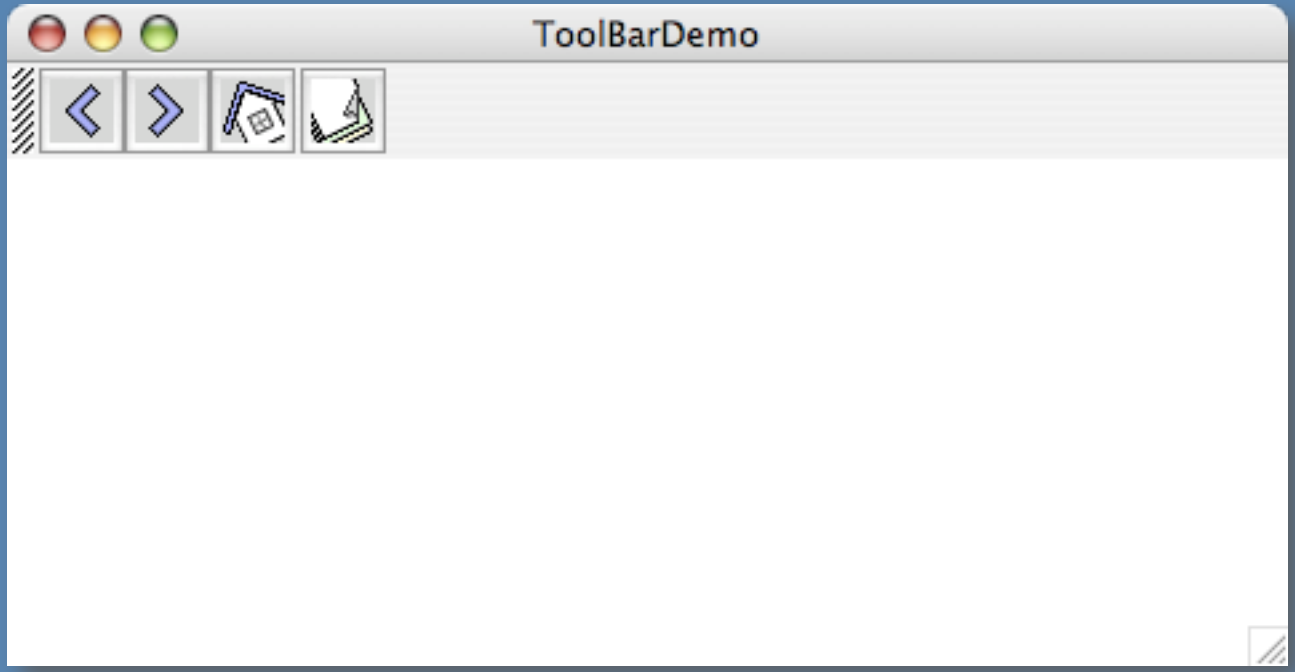


```
import  javax.swing.*;
import  java.awt.*;
import  java.awt.event.*;

public  class  ComboBoxDemo  extends  JFrame  {
     private  JTextArea  textArea  =
                              new  JTextArea(10,30);

   public  static  void  main(String[ ]  args)  {
     ComboBoxDemo  frame  =  new  ComboBoxDemo( );
     frame.pack( );
     frame.setLocation(100,100);
     frame.setVisible(true);
   }
```

```
public  ComboBoxDemo( ) {
   super("ComboBoxDemo");
   addWindowListener(new  WListener( ));
   JComboBox  comboBox  =  new  JComboBox( );

   comboBox.addItem("Apple");
   ...

   comboBox.addActionListener(new  CBListener( ));
   ...

   JPanel  p  =  new  JPanel( );
   p.add(comboBox);
   getContentPane( ).add(p,"East");
   getContentPane( ).add(textArea,"Center");
}
```

```
  class WListener extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
      System.exit(0);
    }
  }

  class CBListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      JComboBox cb = (JComboBox)e.getSource( );
      String flavor =
             (String)cb.getSelectedItem( );
      textArea.append(flavor + "\n");
    }
  }

}
```

ToolBar

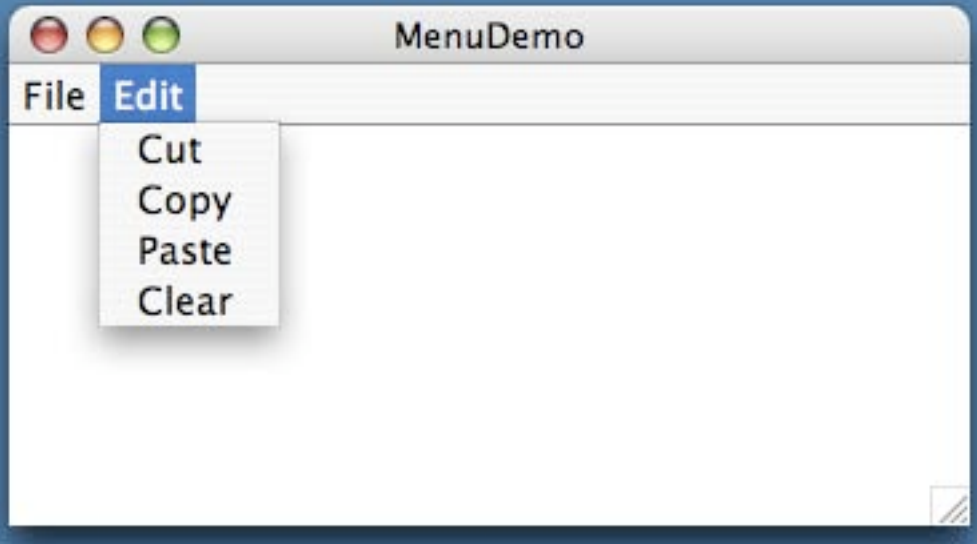


```
public class ToolBarDemo extends JFrame {
    private JTextArea textArea =
                          new JTextArea(10,30);

  public static void main(String[ ] args) {
    ToolBarDemo frame = new ToolBarDemo( );
    ...

  public ToolBarDemo( ) {
    addWindowListener(new WListener( ));
    JToolBar toolBar = createToolBar( );
    ...

    getContentPane( ).add(toolBar, "North");
  }
```

```
protected JToolBar createToolBar( ) {
  JToolBar tb = new JToolBar( );
  JButton button = null;

  button = new JButton(
         new ImageIcon("images/left.gif"));
  button.addActionListener(
         new BListener("left button"));
  tb.add(button);
  ...

  tb.addSeparator( );
  ...

  return tb;
}
```

```
  class BListener implements ActionListener {
    private String actionDescription;

    public BListener(String ad) {
      actionDescription = ad;
    }

    public void actionPerformed(ActionEvent e) {
      textArea.append(actionDescription + "\n");
    }
  }

}
```

MenuBar



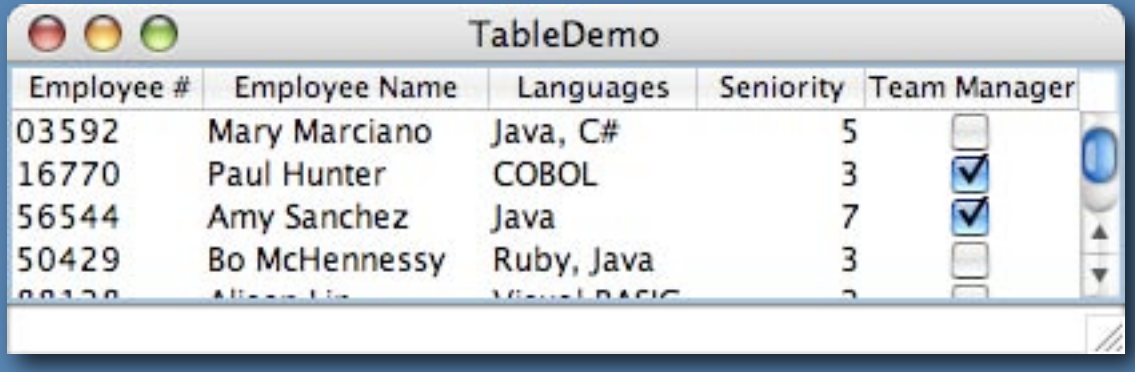
```
public class MenuDemo extends JFrame {
    private static MenuDemo frame =
                          new MenuDemo( );
    private JTextArea textArea =
                          new JTextArea(10,30);
    private ActionListener miListener
        = new MIListener( );

  public static void main(String[ ] args) {...

  public MenuDemo( ) {
    ...
    setJMenuBar(createMenuBar( ));
    ...
  }
```

```
protected JMenuBar createMenuBar( ) {
   JMenuBar mb = new JMenuBar( );
   mb.add(createFileMenu( ));
   mb.add(createEditMenu( ));
   return mb;
}

protected JMenu createFileMenu( ) {
   JMenu menu = new JMenu("File");
   menu.add(createMenuItem("New",miListener));
   menu.add(createMenuItem("Open...",miListener));
   menu.addSeparator( );
   ...
   menu.add(createMenuItem("Quit",
                           new QListener( )));
   return menu;
}
```

```
...
  protected JMenuItem createMenuItem(String s,
                          ActionListener a) {
    JMenuItem menuItem = new JMenuItem(s);
    menuItem.addActionListener(a);
    return menuItem;
}

  class MIListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      JMenuItem mi = (JMenuItem)e.getSource( );
      textArea.append(mi.getText( ) + "\n");
    }
  }
...
```

```
  class QListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      int n = JOptionPane.showConfirmDialog(frame,
          "Are you sure you want to quit?","",
          JOptionPane.YES_NO_OPTION,
          JOptionPane.QUESTION_MESSAGE);
      if (n == JOptionPane.YES_OPTION)
          System.exit(0);
    }
  }

}
```

Table


TableDemo
Employee #
Employee Name
Languages
Seniority
Team Manager
03592
Mary Marciano
Java, C#
5
16770
Paul Hunter
COBOL
3
56544
Amy Sanchez
Java
7
50429
Bo McHennessy
Ruby, Java
3

```
public class TableDemo extends JFrame {
  private JTextField tf = new JTextField(20);

  public static void main(String[] args) {
    ...
  }

  public TableDemo() {
    ...
  }

  class TMListener implements TableModelListener {
    ...
  }

  class MyTableModel extends AbstractTableModel
    ...
  }
}
```

```
public class TableDemo extends JFrame {
  private JTextField tf = new JTextField(20);

  public static void main(String[] args) {
    TableDemo frame = new TableDemo();
    frame.pack();
    frame.setVisible(true);
  }
```

```
public TableDemo() {
  super("TableDemo");

  MyTableModel myModel = new MyTableModel();
  JTable table = new JTable(myModel);
  myModel.addTableModelListener(
                          new TMListener());

  table.setPreferredScrollableViewportSize(
                          new Dimension(400, 70));
  JScrollPane scrollPane = new JScrollPane(table);
  getContentPane().add("Center", scrollPane);

  tf.setFont(new Font("Helvetica", Font.BOLD, 12));
  getContentPane().add("South", tf);
}
```

```
class TMListener implements TableModelListener {

  public void tableChanged(TableModelEvent e) {
    int row = e.getFirstRow();
    int column = e.getColumn();
    TableModel model = (TableModel)e.getSource();
    String colName = model.getColumnName(column);
    Object ID = model.getValueAt(row, 0);
    Object val = model.getValueAt(row, column);
    tf.setText(colName + "[" + ID + "] = " + val);
  }
}
```

```
class MyTableModel extends AbstractTableModel {
  final String[] columnNames = ...
  final Object[][] data = ...

  public int getColumnCount() {...

  public int getRowCount() {...

  public Object getValueAt(int r, int c) {...

  public void setValueAt(Object v, int r, int c) {...

  public String getColumnName(int c) {...

  public Class getColumnClass(int c) {...

  public boolean isCellEditable(int r, int c) {...
}
```

```
class MyTableModel extends AbstractTableModel {

  final String[] columnNames =
    {"Employee #", "Employee Name", "Languages",
          "Seniority", "Team Manager"};

  final Object[][] data = {
    {"03592", "Mary Marciano", "Java, C#",
          new Integer(5), new Boolean(false)},
    ...
    {"95543", "Rob Helgerson", "APL",
          new Integer(4), new Boolean(false)}
  };

  ...
```

```
public int getColumnCount() {
  return columnNames.length;
}

public int getRowCount() {
  return data.length;
}

public Object getValueAt(int r, int c) {
  return data[r][c];
}

public void setValueAt(Object v, int r, int c) {
  data[r][c] = v;
  fireTableCellUpdated(r, c);
}
...
```

```
  public String getColumnName(int c) {
    return columnNames[c];
  }

  //used to determine renderer/editor per cell.
  public Class getColumnClass(int c) {
    return getValueAt(0, c).getClass();
  }

  public boolean isCellEditable(int r, int c) {
    return ((c == 2) | (c == 4));
  }

}     //end of MyTableModel
```

Java graphics & event-driven UIs

- Java event-delegation model
- Swing highlights
- Swing components survey
- ⇒ Java graphics

Basics of Java graphics

The `Graphics` class: the 'cornerstone' of Java graphics

An abstract class with mostly abstract methods that gets extended by classes you won't see

Its objects are the 'graphics contexts' for the drawing surfaces of components

public abstract class Graphics

---

drawLine(int,int,int,int)

drawRect(int,int,int,int)

drawOval(int,int,int,int)

drawPolygon(Polygon)

drawRoundRect(int,int,int,int,int,int)

drawPolyline(int[],int[],int)

drawArc(int,int,int,int,int,int)

drawString(String,int,int)

drawImage(Image,int,int,ImageObserver)

```
fillRect(int,int,int,int)
fillOval(int,int,int,int)
fillPolygon(Polygon)
fillRoundRect(int,int,int,int,int,int)
fillArc(int,int,int,int,int,int)

clearRect(int,int,int,int)

getColor()
setColor(Color)
```

`getClip()` Gets the current clipping area

`getClipBounds()` Gets bounding rectangle of clip

`setClip(Shape)` Sets clip to an arbitrary shape

`setClip(int,int,int,int)` …to a rectangle

`clipRect(int,int,int,int)`
Intersects clip with the specified rectangle

Clipping area:

region within which drawing is enabled

`Graphics` ghostly presence

You never create `Graphics` objects

Your program's flow of control doesn't execute `Graphics` methods directly

But you can arrange to have the methods you choose be executed

Painting

AWT's `Component` has a `paint` method

It gets called automagically in two cases:

1. it needs doing
2. you've asked for it to be done
   (by calling `repaint`)

`Jcomponent` overrides `paint`

Painting hierarchy

`JComponent.paint` calls

- `paintComponent`
- `paintBorder`
- `paintChildren`

`paintChildren` calls the `paint` method of each component contained

Component painting

`paintComponent` calls the methods of the component's UI delegate to draw the component itself

It is passed a copy of the `Graphics` object:

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

Custom painting

So...

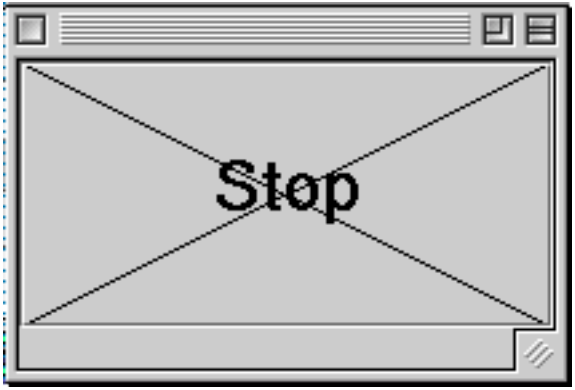
you can extend a component

and override `paintComponent`

Use `super.paintComponent` to draw it,

then whatever you want

to draw right on the component


Stop

```
class XButton extends JButton {

  public XButton(String s) {super(s);}

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    int xx = getWidth() - 1;
    int yy = getHeight() - 1;
    g.drawLine(0,0,xx,yy);
    g.drawLine(0,yy,xx,0);
  }
}
```

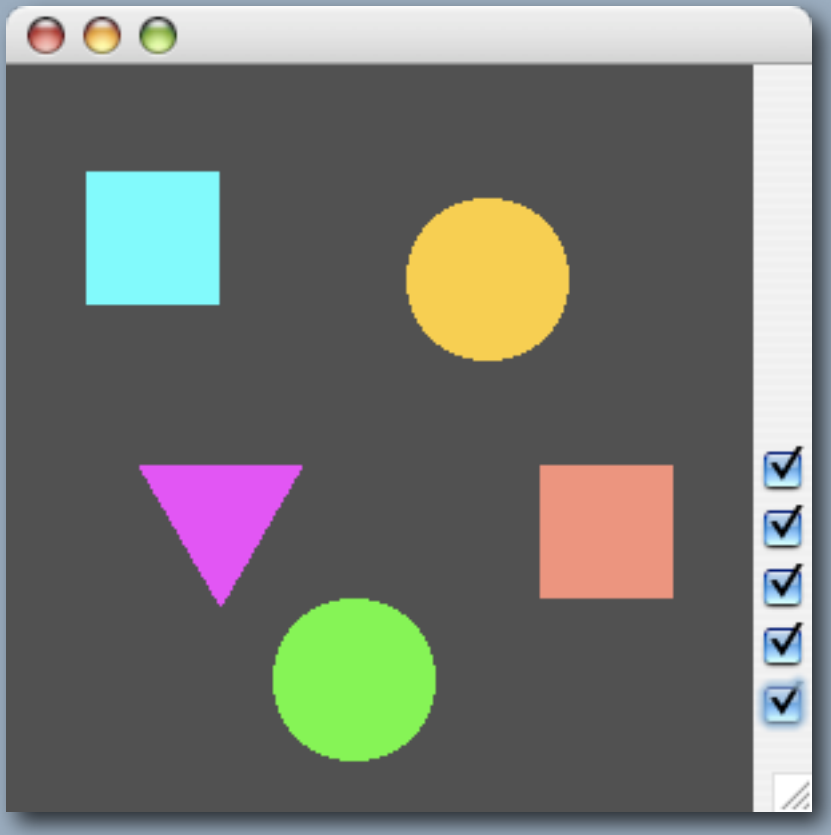
Custom painting

Usually, you choose a plain component to draw on, e.g., a `JPanel`

Should still call `super.paintComponent` so background gets drawn properly

Otherwise, never call `paintComponent` yourself
Call `repaint` instead (But never override `repaint`)

Graphics demo



```
public class ShapeShow extends JFrame {
  AbstractShape[] sh = ...

  public static void main(String a[]) {
    new ShapeShow();
  }

  public ShapeShow() {
    ...
  }

  class CBListener implements ItemListener { {
    ...
  }
}
```

plus classes ShapePanel, AbstractShape, Square etc.

```
sh = { new  Square(30, 40, Color.cyan),
       new  Circle(150, 50, Color.orange),
       new  Square(200, 150, new  Color(255, 125, 100)),
       new  Triangle(50, 150, Color.magenta),
       new  Circle(100, 200, Color.green)  }
```

```
public ShapeShow() {
  ShapePanel sp = new ShapePanel(sh, 280, 280);
  getContentPane().add("Center", sp);

  Box cbp = new Box(BoxLayout.Y_AXIS);
  cbp.add(Box.createGlue());
  CBListener cbl = new CBListener(sp);
  for (int i = 0; i < sh.length; i++) {
    sh[i].addItemListener(cbl);
    cbp.add(sh[i]);
  }
  cbp.add(Box.createVerticalStrut(30));
  getContentPane().add("East", cbp);
  ...
}
```

```
abstract class AbstractShape extends JCheckBox {
  protected int x, y;
  protected Color c;

  public AbstractShape(int xx, int yy, Color cc) {
    x = xx; y = yy; c = cc;
  }

  public boolean checked( ) {
    return (getSelectedObjects( ) != null);
  }

  public abstract void draw(Graphics x) ;
}
```

```
class Triangle extends AbstractShape {

   public Square(int xx, int yy, Color cc) {
      super(xx, yy, cc);
   }

   public void draw(Graphics gx) {
      gx.setColor(c);
      Polygon p = new Polygon();
      p.addPoint(x, y);
      p.addPoint(x+30, y+52);
      p.addPoint(x+60, y);
      gx.fillPolygon(p);
   }
}
```

square: `gx.fillRect(x, y, 50, 50)`

circle: `gx.fillOval(x, y, 60, 60)`

```
class ShapePanel extends JPanel {
  private AbstractShape[] sh = null;

  public ShapePanel(AbstractShape[] shapes, int w, int h) {
    sh = shapes;
    setPreferredSize(new Dimension(w, h));
    setBackground(Color.darkGray);
  }

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    for (int i = 0; i < sh.length; i++)
      if (sh[i].checked()) {
        sh[i].draw(g);
      }
  }
}
```

```
class CBListener implements ItemListener {
  private JComponent canvas;

  public CBListener(JComponent c) {canvas = c;}

  public void itemStateChanged(ItemEvent e) {
    canvas.repaint();
  }
}
```