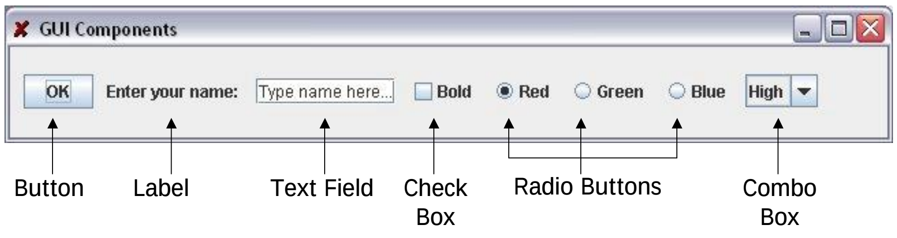
Java GUI Programming

Sean P. Strout (sps@cs.rit.edu)
Robert Duncan (rwd@cs.rit.edu)

What is a GUI?

- Java has standard packages for creating custom **Graphical User Interfaces**
- Some of the fundamental GUI components:



ShowComponents.java

```
import javax.swing.*;
import java.awt.*;
public class ShowComponents extends JFrame {
        public ShowComponents () {
                Container container = getContentPane();
                container.setLayout(new FlowLayout (FlowLayout.LEFT, 10, 20));
                container.add(new JButton("OK"));
                container.add(new JLabel("Enter your name: "));
                container.add(new JTextField("Type name here..."));
                container.add(new JCheckBox("Bold"));
                container.add(new JRadioButton("Red", true));
                container.add(new JRadioButton("Green"));
                container.add(new JRadioButton("Blue"));
                container.add(new JComboBox(new String[]{"High", "Med", "Low"}));
        } // ShowComponents
        public static void main(String args[]) {
                ShowComponents frame = new ShowComponents();
                frame.setTitle("GUI Components");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setSize(650, 100);
                frame.setLocation(100, 100);
                frame.setVisible(true);
        } // main
} // ShowComponents
```

What is AWT?

- The **Abstract Window Toolkit** was a part of Java from the beginning

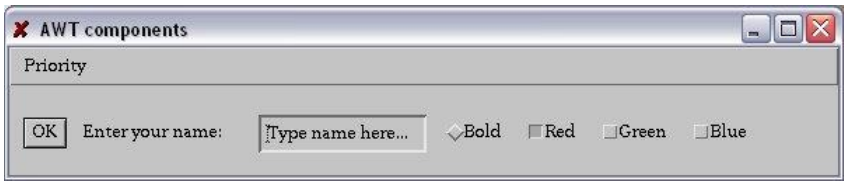
  ```
  import java.awt.*;
  ```

- All AWT components must be mapped to platform specific components using **peers**
  - The look and feel of these components is tied to the native components of the window manager

- AWT components are considered to be very error prone and should not be used in modern Java applications

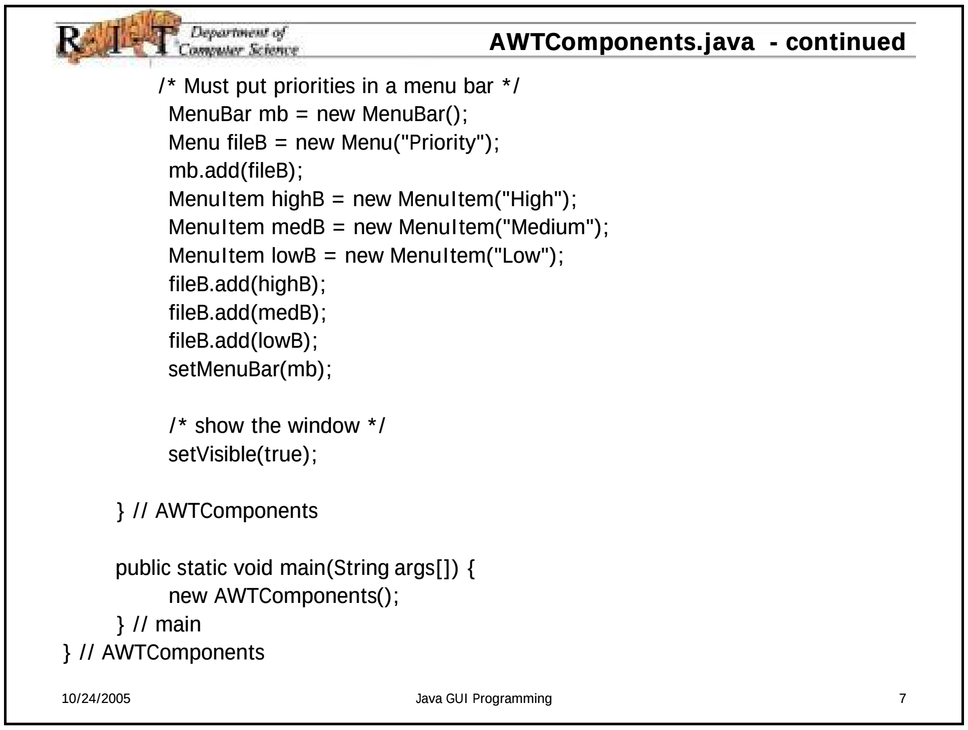
What is AWT?

- The same application using only AWT components, running on X-Windows:



AWTComponents.java

```
import java.awt.*;
public class AWTComponents extends Frame {
        public AWTComponents () {
                /* set up the window layout */
                setLayout(new FlowLayout (FlowLayout.LEFT, 10, 20));
                setLocation(100, 100);
                setTitle("AWT components");
                resize(600, 125);

                /* button */
                add(new Button("OK"));
                /* label */
                add(new Label("Enter your name: "));
                /* text field */
                add(new TextField("Type name here..."));

                /* check box group */
                CheckboxGroup cbg = new CheckboxGroup();
                add(new Checkbox("Bold", cbg, false));
                add(new Checkbox("Red", null, true));
                add(new Checkbox("Green"));
                add(new Checkbox("Blue"));
```

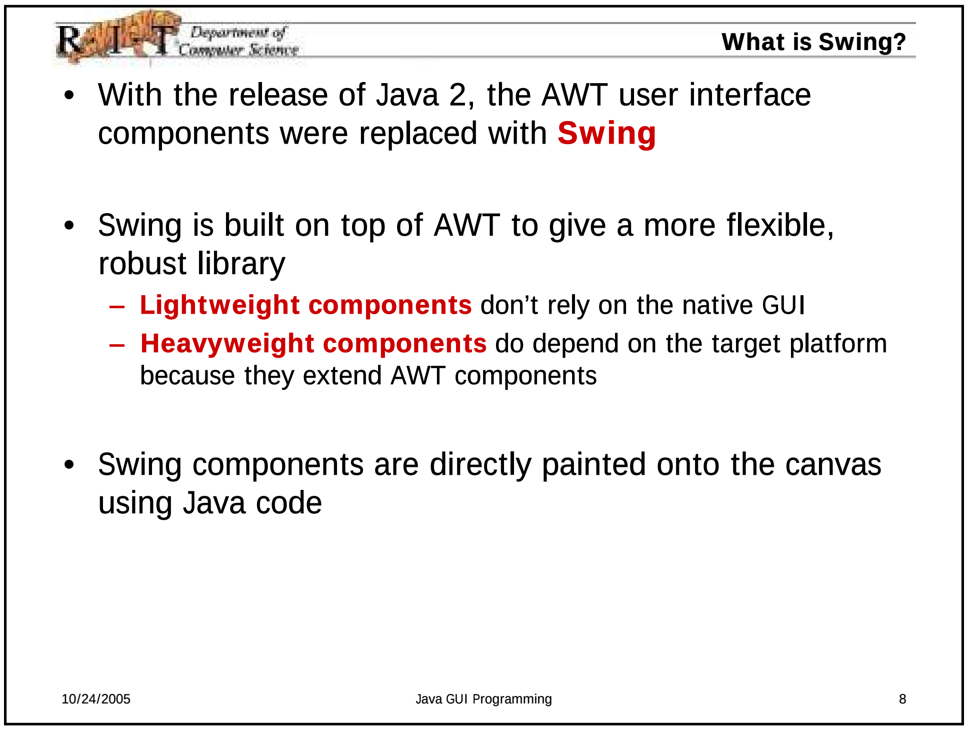


Department of Computer Science

AWTComponents.java - continued

```
        /* Must put priorities in a menu bar */
         MenuBar mb = new MenuBar();
         Menu fileB = new Menu("Priority");
         mb.add(fileB);
         MenuItem highB = new MenuItem("High");
         MenuItem medB = new MenuItem("Medium");
         MenuItem lowB = new MenuItem("Low");
         fileB.add(highB);
         fileB.add(medB);
         fileB.add(lowB);
         setMenuBar(mb);

         /* show the window */
         setVisible(true);

    } // AWTComponents

    public static void main(String args[]) {
         new AWTComponents();
    } // main
} // AWTComponents
```

10/24/2005 Java GUI Programming 7



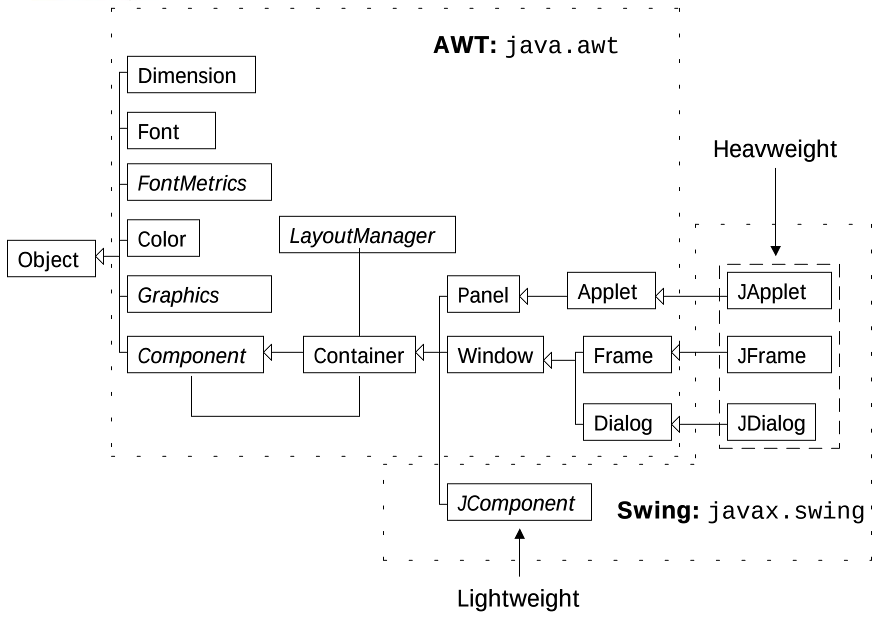
Department of Computer Science

What is Swing?

- With the release of Java 2, the AWT user interface components were replaced with **Swing**
- Swing is built on top of AWT to give a more flexible, robust library
  - **Lightweight components** don't rely on the native GUI
  - **Heavyweight components** do depend on the target platform because they extend AWT components
- Swing components are directly painted onto the canvas using Java code

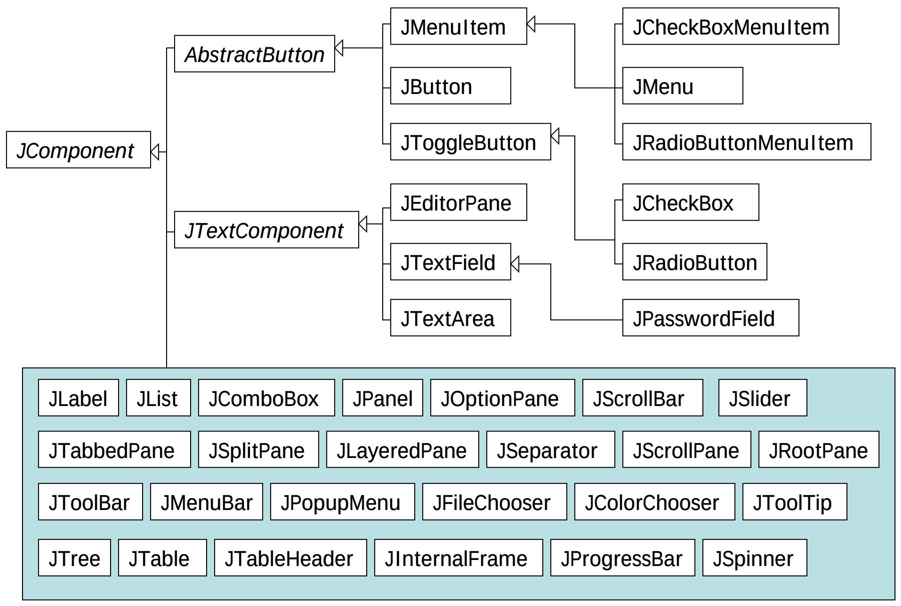
10/24/2005 Java GUI Programming 8

Java GUI API



10/24/2005 Java GUI Programming 9

Java GUI API



10/24/2005 Java GUI Programming 10

Department of Computer Science

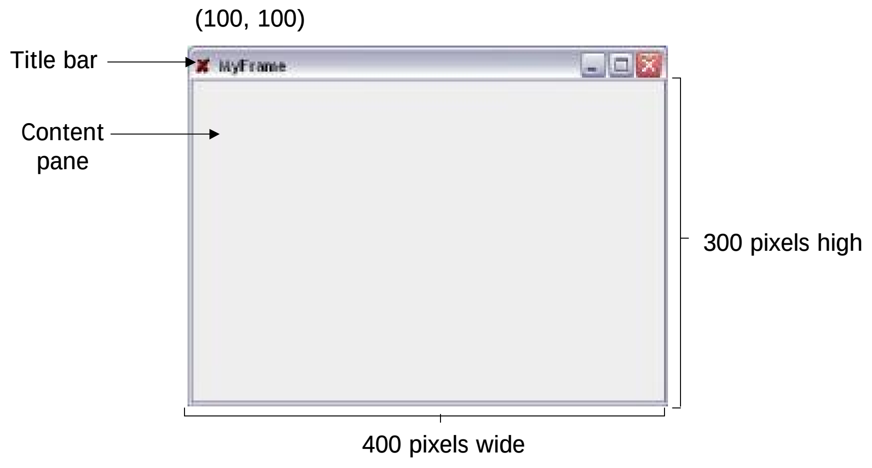
Container Classes

- **Container classes** are GUI components that are used as containers to contain other GUI components
  - For Swing use: `Component, Container, JFrame, JDialog, JApplet, Jpanel`
  - **`JFrame`** is a window not contained inside another window
  - **`JDialog`** is a temporary popup window or message box
  - **`JApplet`** is an applet that can run in a web browser
  - **`JPanel`** is an invisible, nest-able container used to hold UI components or canvases to draw graphics
- A **layout manager** is used to position and place components in a container

Department of Computer Science

Frames

- You need a **frame** to hold the UI components



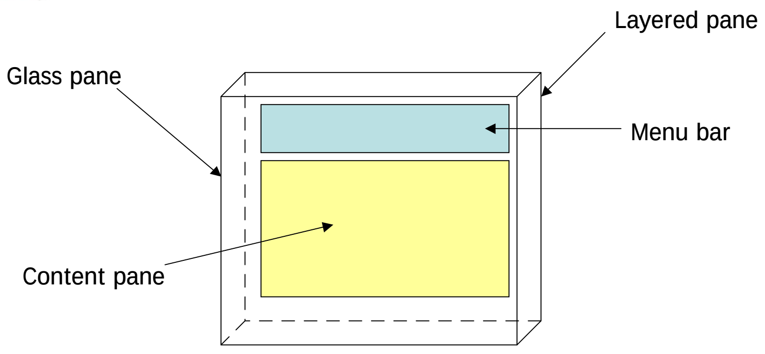
Department of Computer Science

MyFrame.java

```
import javax.swing.*;
public class MyFrame {
	public static void main(String args[]) {
		// Create frame with the title "MyFrame"
		JFrame frame = new JFrame("MyFrame");

		// Set the size of the frame to width=400, height=300.
		// If this is not set, it will just be the size of the title bar
		frame.setSize(400, 300);

		// The frame is not displayed until this statement
		frame.setVisible(true);

		// The location of the frame on the screen.  The upper-left corner is 0,0
		frame.setLocation(100, 100);

		// Tell the program to terminate when the frame is closed.
		// If this is not used, the program does not terminate and
		// it must be stopped manually (CTRL-Z then 'kill %' )
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	} // main
} // MyFrame
```

Department of Computer Science

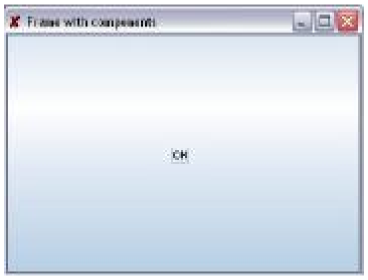
Content Pane



- The **content pane** is the part of the frame where the UI components will be placed
- It is a `java.awt.Container` object

Adding Components to a Frame

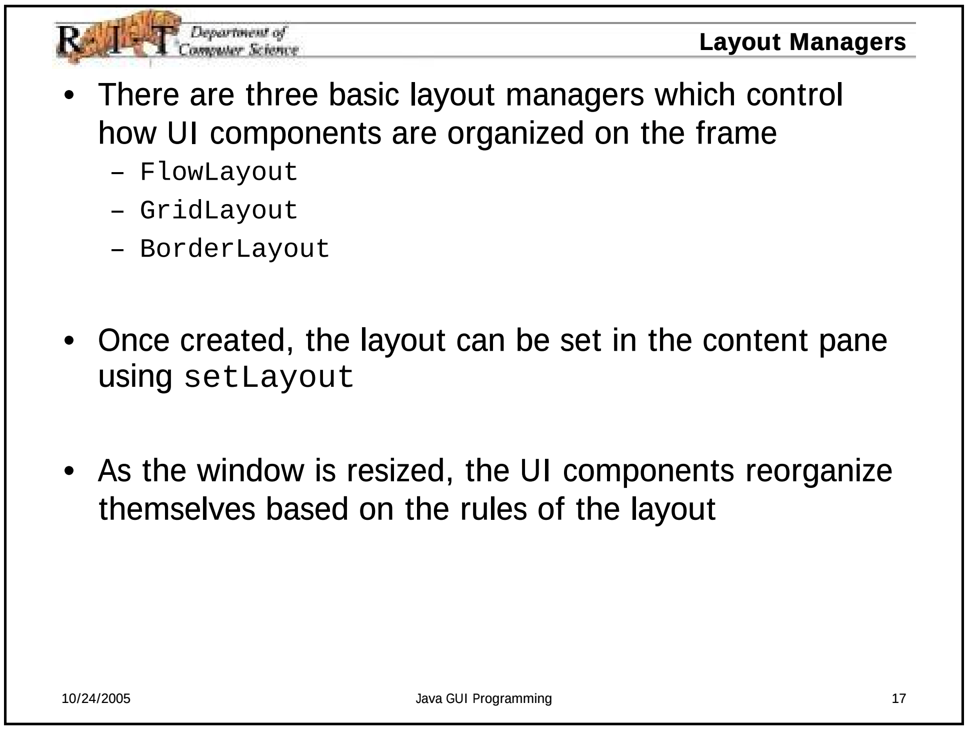
- UI components can be add'ed to the content pane after they are created



- Here, the OK button is centered in the frame and occupies the whole frame, no matter how it is resized

MyFrameWithButton.java

```
import javax.swing.*;           // JFrame
import java.awt.*;              // Container
public class MyFrameWithButton {
        public static void main(String args[]) {
                JFrame frame = new JFrame("Frame with components");

                // Get the content pane, which was made when the
                // frame was created.
                Container container = frame.getContentPane();
                // Create an "OK" button
                JButton okButton = new JButton("OK");
                // Add the button into the frame via the content pane
                container.add(okButton);

                // Set up the frame behavior
                frame.setSize(400, 300);
                frame.setLocation(100, 100);
                frame.setVisible(true);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        } // main
} // MyFrameWithButton
```



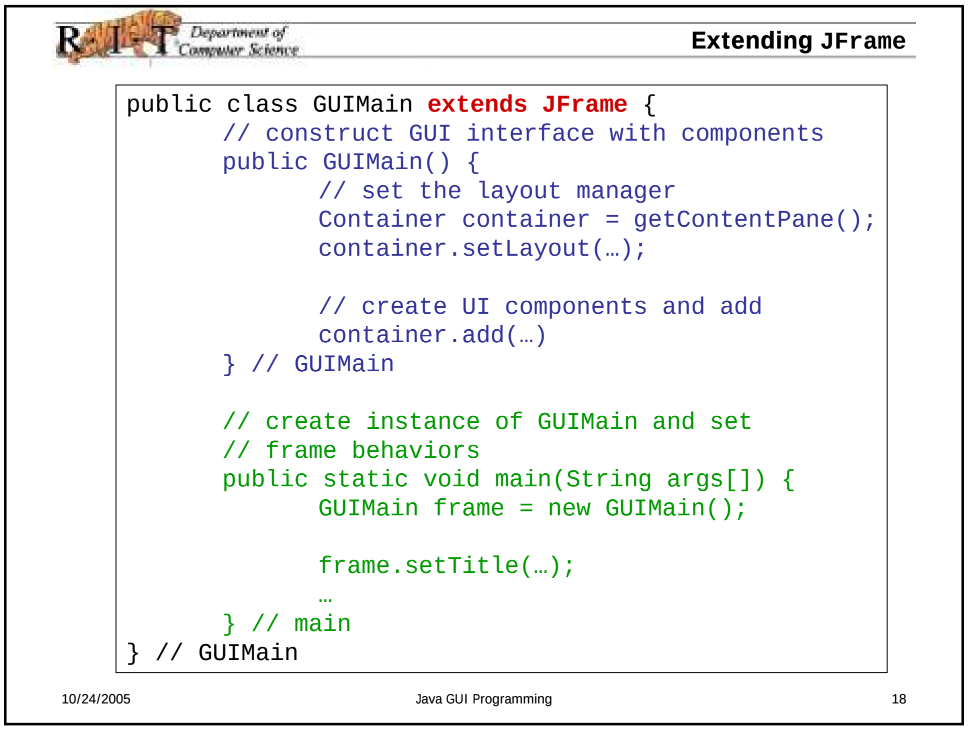
Layout Managers

- There are three basic layout managers which control how UI components are organized on the frame
  - `FlowLayout`
  - `GridLayout`
  - `BorderLayout`

- Once created, the layout can be set in the content pane using `setLayout`

- As the window is resized, the UI components reorganize themselves based on the rules of the layout

10/24/2005 Java GUI Programming 17



Extending `JFrame`

```
public class GUIMain extends JFrame {
       // construct GUI interface with components
       public GUIMain() {
              // set the layout manager
              Container container = getContentPane();
              container.setLayout(…);

              // create UI components and add
              container.add(…)
       } // GUIMain

       // create instance of GUIMain and set
       // frame behaviors
       public static void main(String args[]) {
              GUIMain frame = new GUIMain();

              frame.setTitle(…);
              …
       } // main
} // GUIMain
```

10/24/2005 Java GUI Programming 18

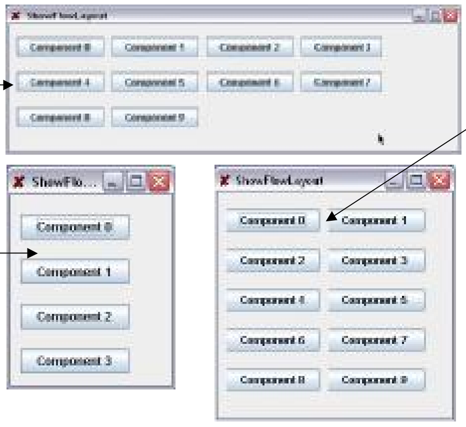
FlowLayout

- With **flow layout**, the components arrange themselves from left to right in the order they were added

Rows/buttons are left aligned using `FlowLayout.LEFT`

Horizontal gap of 10 pixels

Vertical gap of 20 pixels



ShowFlowLayout.java

```
public class ShowFlowLayout extends JFrame {
    // Constructor places components in the frame
    public ShowFlowLayout() {
        // Get the content pane from the frame
        Container container = getContentPane();
        // Set FlowLayout, aligned left with a horizontal
        // gap 10 and vertical gap 20 between components
        container.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));
        // Add 10 buttons into the frame
        for (int i=0; i<10; i++) {
            container.add(new JButton("Component " + i));
        }
    } // ShowFlowLayout
    public static void main(String args[]) {
        ShowFlowLayout frame = new ShowFlowLayout();
        frame.setTitle("ShowFlowLayout");
        frame.setSize(600, 200);
        frame.setLocation(100, 100);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    } // main
} // ShowFlowLayout
```

Department of Computer Science

GridLayout

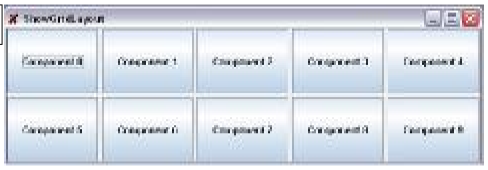
- With **grid layout**, the components arrange themselves in a matrix formation (rows, columns)
- Either the row or column must be non-zero
- The non-zero dimension is fixed and the zero dimension is determined dynamically
- The dominating parameter is the rows

10/24/2005 Java GUI Programming 21

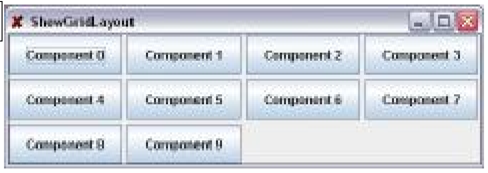
Department of Computer Science

GridLayout

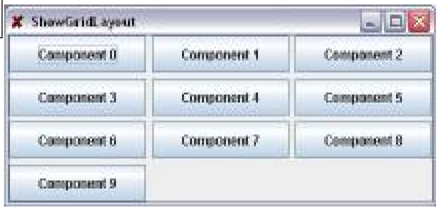
2,4



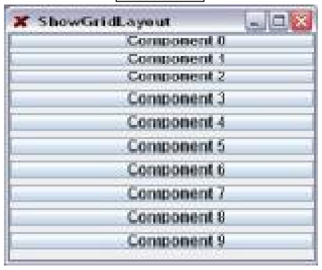
0,4



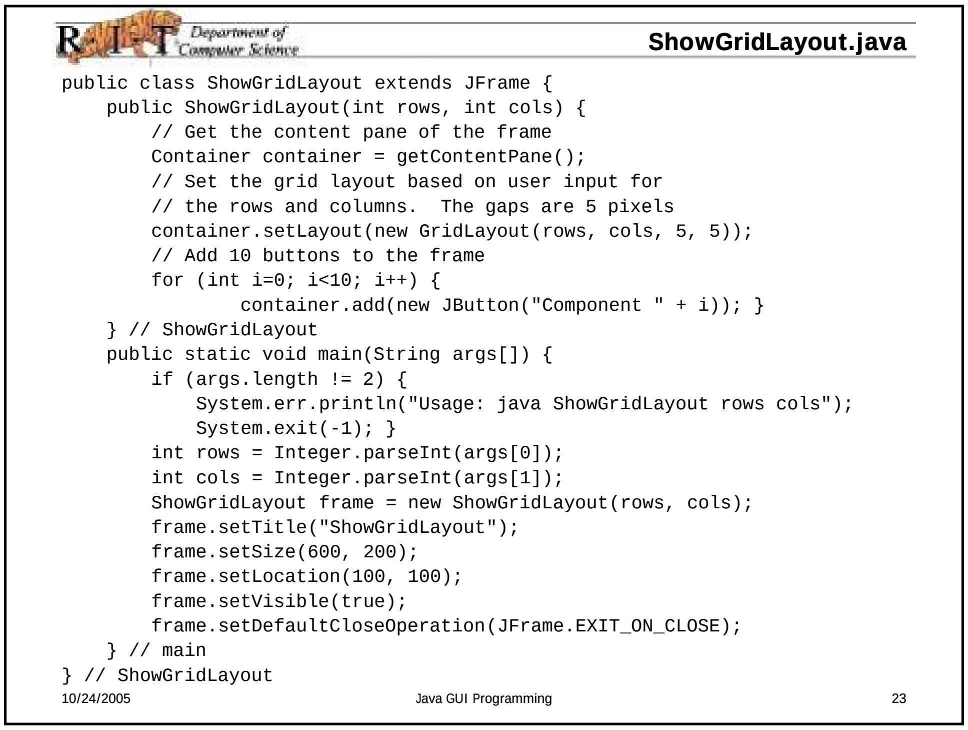
4,4



10, 10



10/24/2005 Java GUI Programming 22

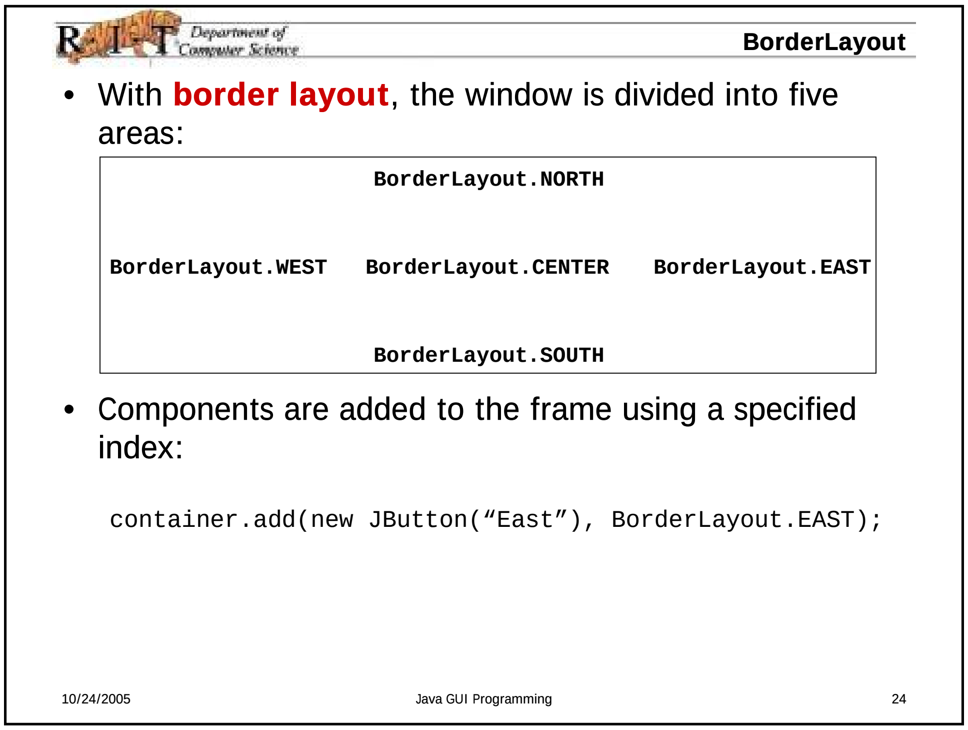


RIT Department of Computer Science

ShowGridLayout.java

```
public class ShowGridLayout extends JFrame {
    public ShowGridLayout(int rows, int cols) {
        // Get the content pane of the frame
        Container container = getContentPane();
        // Set the grid layout based on user input for
        // the rows and columns.  The gaps are 5 pixels
        container.setLayout(new GridLayout(rows, cols, 5, 5));
        // Add 10 buttons to the frame
        for (int i=0; i<10; i++) {
                container.add(new JButton("Component " + i)); }
    } // ShowGridLayout
    public static void main(String args[]) {
        if (args.length != 2) {
            System.err.println("Usage: java ShowGridLayout rows cols");
            System.exit(-1); }
        int rows = Integer.parseInt(args[0]);
        int cols = Integer.parseInt(args[1]);
        ShowGridLayout frame = new ShowGridLayout(rows, cols);
        frame.setTitle("ShowGridLayout");
        frame.setSize(600, 200);
        frame.setLocation(100, 100);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    } // main
} // ShowGridLayout
```

10/24/2005 Java GUI Programming 23



RIT Department of Computer Science

BorderLayout

- With **border layout**, the window is divided into five areas:

BorderLayout.NORTH

BorderLayout.WEST **BorderLayout.CENTER** **BorderLayout.EAST**

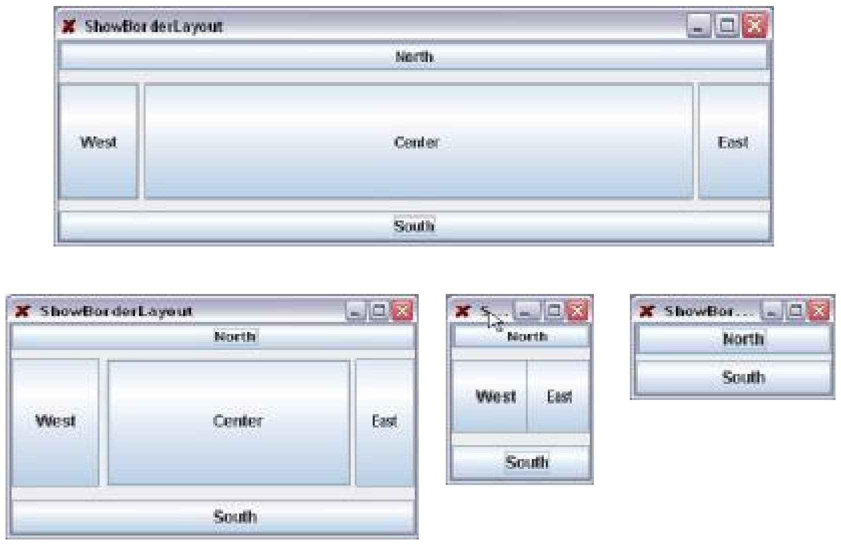
BorderLayout.SOUTH

- Components are added to the frame using a specified index:

```
container.add(new JButton(“East”), BorderLayout.EAST);
```

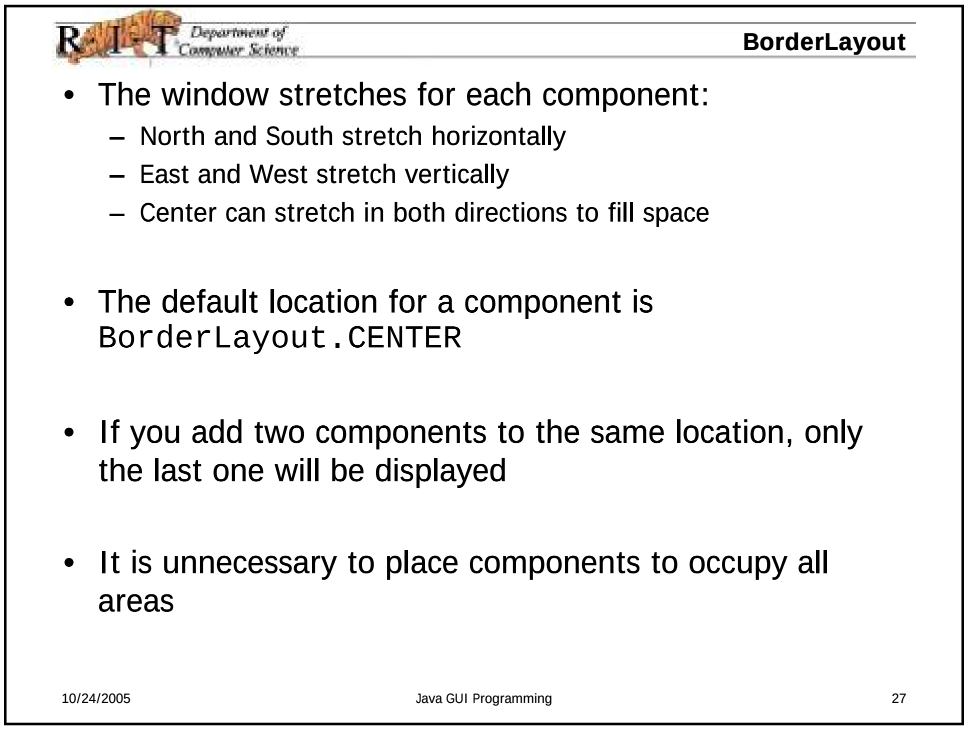
10/24/2005 Java GUI Programming 24

BorderLayout



ShowBorderLayout.java

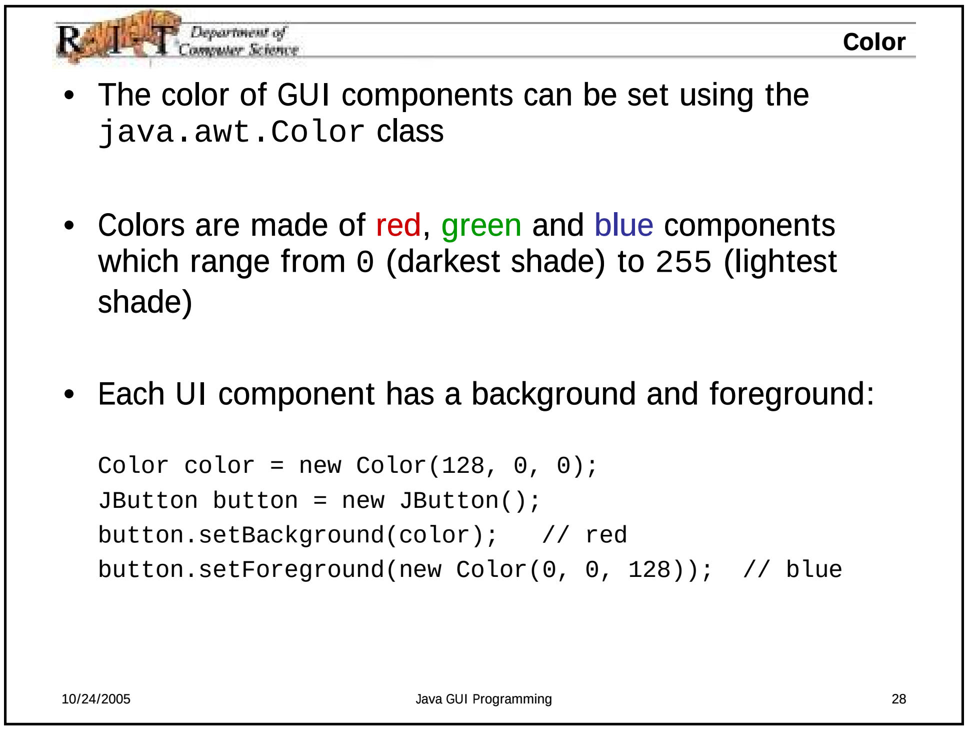
```
public class ShowBorderLayout extends JFrame {
        public ShowBorderLayout() {
                // Get the content pane of the frame
                Container container = getContentPane();
                // Set the border layout with horizontal gap 5
                // and vertical gap 10
                container.setLayout(new BorderLayout(5, 10));
                // Add buttons to the frame
                container.add(new JButton("East"), BorderLayout.EAST);
                container.add(new JButton("South"), BorderLayout.SOUTH);
                container.add(new JButton("West"), BorderLayout.WEST);
                container.add(new JButton("North"), BorderLayout.NORTH);
                container.add(new JButton("Center"), BorderLayout.CENTER);
        } // ShowBorderLayout
        public static void main(String args[]) {
                ShowBorderLayout frame = new ShowBorderLayout();
                frame.setTitle("ShowBorderLayout");
                frame.setSize(600, 200);
                frame.setLocation(100, 100);
                frame.setVisible(true);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        } // main
} // ShowBorderLayout
```



BorderLayout

- The window stretches for each component:
  - North and South stretch horizontally
  - East and West stretch vertically
  - Center can stretch in both directions to fill space
- The default location for a component is `BorderLayout.CENTER`
- If you add two components to the same location, only the last one will be displayed
- It is unnecessary to place components to occupy all areas

10/24/2005 Java GUI Programming 27



Color

- The color of GUI components can be set using the `java.awt.Color` class
- Colors are made of red, green and blue components which range from `0` (darkest shade) to `255` (lightest shade)
- Each UI component has a background and foreground:

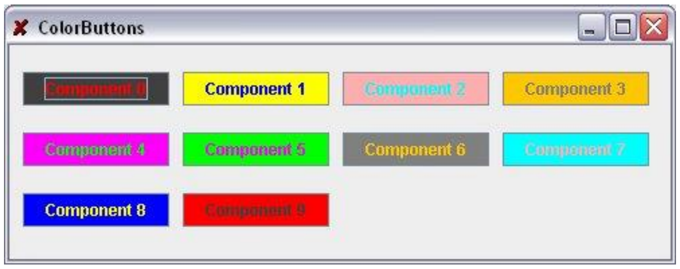
```
Color color = new Color(128, 0, 0);
JButton button = new JButton();
button.setBackground(color);   // red
button.setForeground(new Color(0, 0, 128));  // blue
```

10/24/2005 Java GUI Programming 28

RIT Department of Computer Science

Color

- There are 13 constant colors defined in `Color`:
  - `BLACK, BLUE, CYAN, DARK_GRAY, GRAY, GREEN, LIGHT_GRAY, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW`



RIT Department of Computer Science

ColorButtons.java

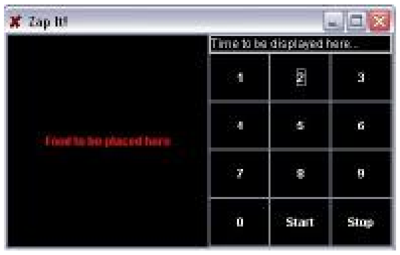
```
public class ColorButtons extends JFrame {

        // Constructor places components in the frame
        public ColorButtons() {
                // Get the content pane from the frame
                Container container = getContentPane();

                // Set FlowLayout, aligned left with a horizontal
                // gap 10 and vertical gap 20 between components
                container.setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));

                Color colors[] = new Color [10];
                colors[0] = Color.RED;
                colors[1] = Color.BLUE;
                colors[2] = Color.CYAN;
                colors[3] = Color.GRAY;
                colors[4] = Color.GREEN;
                colors[5] = Color.MAGENTA;
                colors[6] = Color.ORANGE;
                colors[7] = Color.PINK;
                colors[8] = Color.YELLOW;
                colors[9] = Color.DARK_GRAY;
```

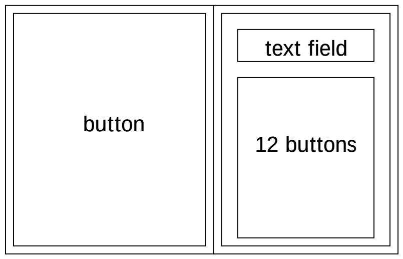
ColorButtons.java

```
             // Add 10 buttons into the frame
             for (int i=0; i<10; i++) {
                    JButton button = new JButton("Component " + i);
                    button.setForeground(colors[i]);
                    button.setBackground(colors[9-i]);
                    container.add(button);
             }
      } // ColorButtons

      public static void main(String args[]) {
             ColorButtons frame = new ColorButtons();
             frame.setTitle("ColorButtons");
             frame.setSize(600, 200);
             frame.setLocation(100, 100);
             frame.setVisible(true);
             frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      } // main
} // ColorButtons
```

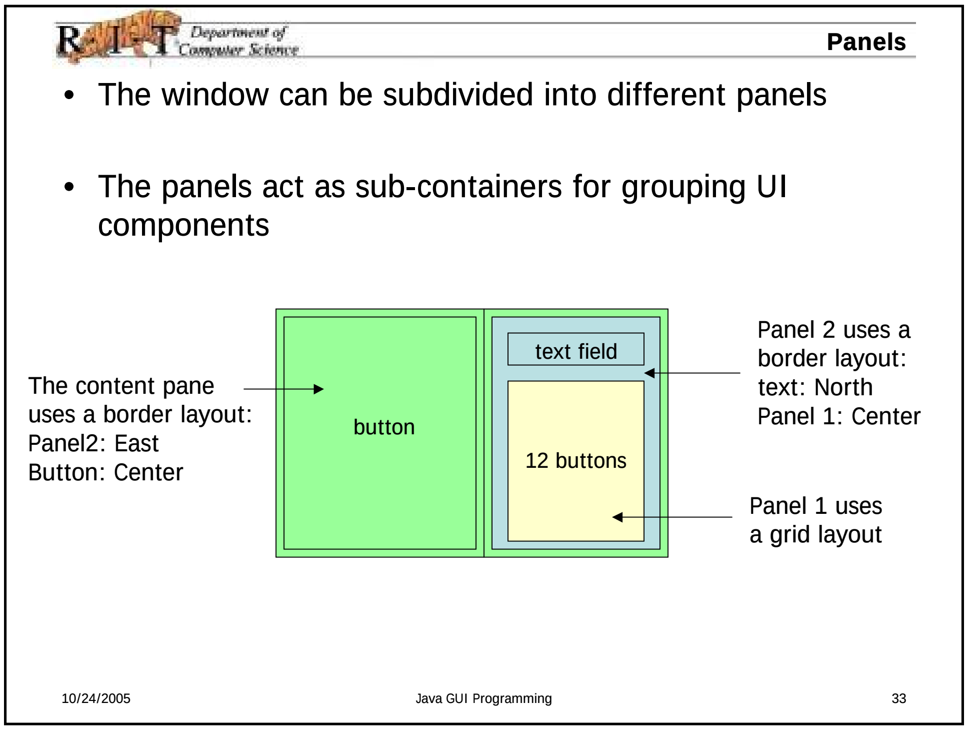
Panels

- Write a program to organize the components for a microwave oven:



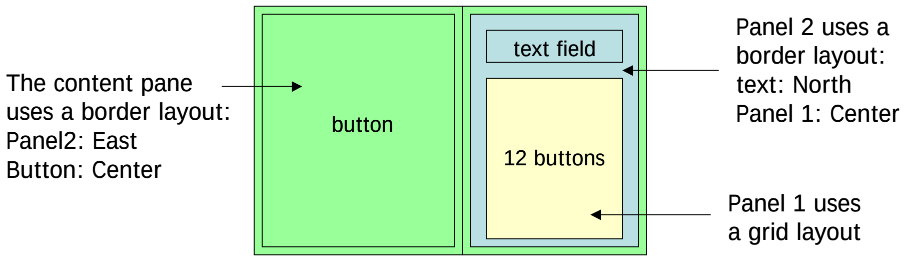


- The problem is we want to use different layouts for different components

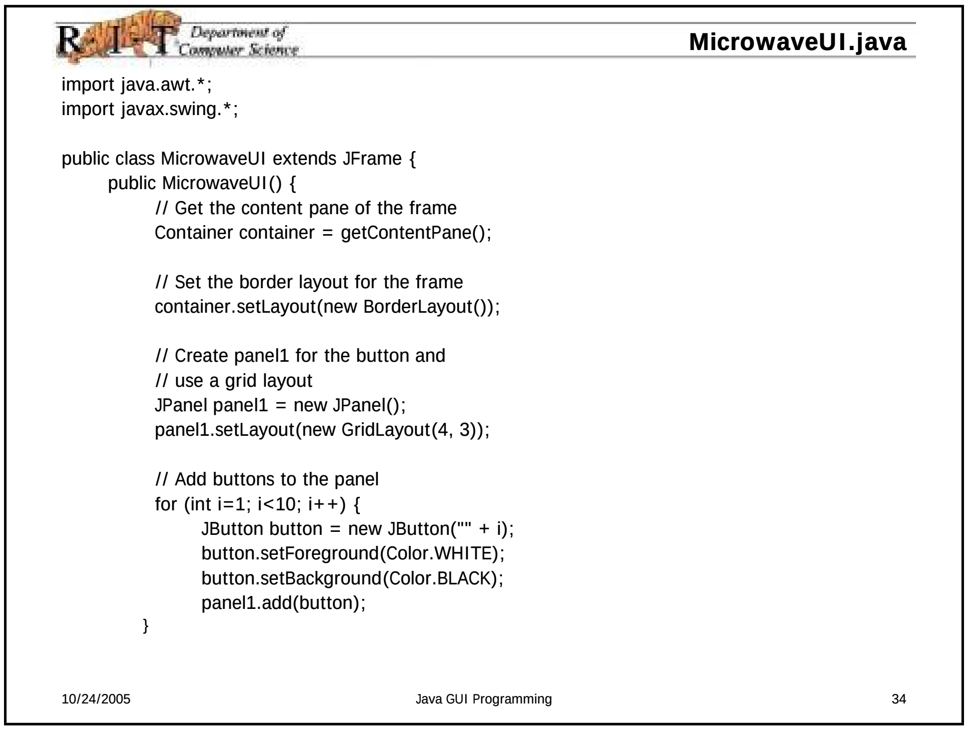


Panels

- The window can be subdivided into different panels
- The panels act as sub-containers for grouping UI components


The content pane uses a border layout:
Panel2: East
Button: Center
button
text field
12 buttons
Panel 2 uses a border layout:
text: North
Panel 1: Center
Panel 1 uses a grid layout

10/24/2005 Java GUI Programming 33



MicrowaveUI.java

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

public class MicrowaveUI extends JFrame {
        public MicrowaveUI() {
                // Get the content pane of the frame
                Container container = getContentPane();

                // Set the border layout for the frame
                container.setLayout(new BorderLayout());

                // Create panel1 for the button and
                // use a grid layout
                JPanel panel1 = new JPanel();
                panel1.setLayout(new GridLayout(4, 3));

                // Add buttons to the panel
                for (int i=1; i<10; i++) {
                        JButton button = new JButton("" + i);
                        button.setForeground(Color.WHITE);
                        button.setBackground(Color.BLACK);
                        panel1.add(button);
              }
```

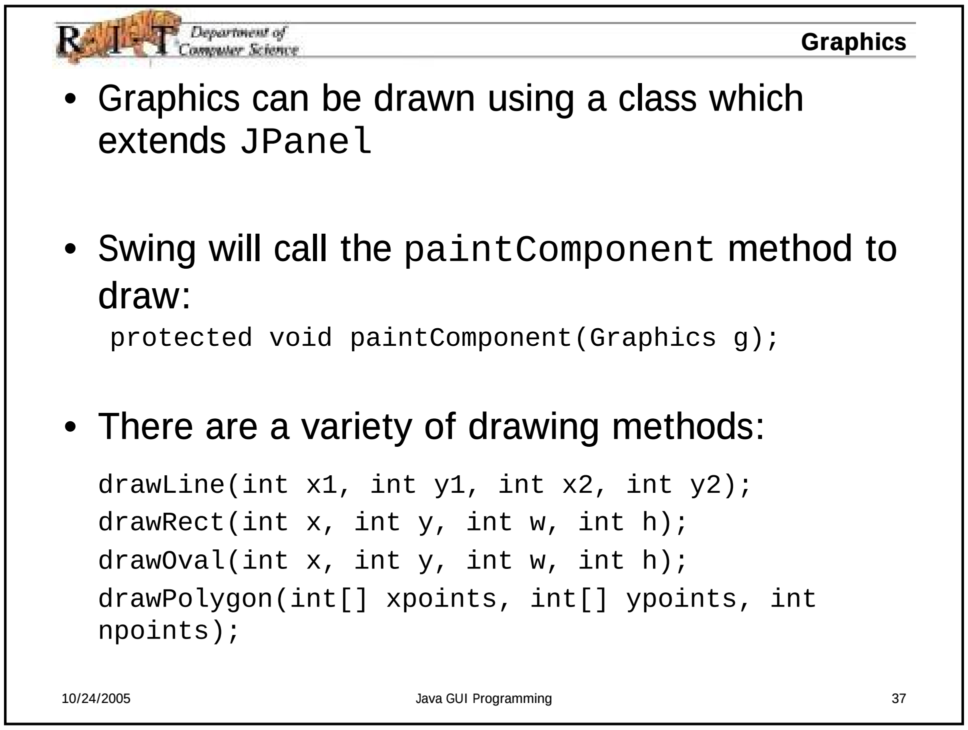
10/24/2005 Java GUI Programming 34

MicrowaveUI.java

```
        JButton button = new JButton("0");
        button.setForeground(Color.WHITE);
        button.setBackground(Color.BLACK);
        panel1.add(button);
        button = new JButton("Start");
        button.setForeground(Color.WHITE);
        button.setBackground(Color.BLACK);
        panel1.add(button);
        button = new JButton("Stop");
        button.setForeground(Color.WHITE);
        button.setBackground(Color.BLACK);
        panel1.add(button);

        // Create panel2 to hold a text field and panel1
        JPanel panel2 = new JPanel(new BorderLayout());
        JTextField textField =
            new JTextField("Time to be displayed here...");
        textField.setForeground(Color.WHITE);
        textField.setBackground(Color.BLACK);
        panel2.add(textField,BorderLayout.NORTH);
        panel2.add(panel1, BorderLayout.CENTER);
        // Add panel2 and a button to the frame
        container.add(panel2, BorderLayout.EAST);
        button = new JButton("Food to be placed here");
        button.setForeground(Color.RED);
        button.setBackground(Color.BLACK);
        container.add(button, BorderLayout.CENTER);

    } // MicrowaveUI
```

MicrowaveUI.java

```
    public static void main(String args[]) {
        MicrowaveUI frame = new MicrowaveUI();
        frame.setTitle("Zap It!");
        frame.setSize(400, 250);
        frame.setLocation(100, 100);
        frame.setVisible(true);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    } // main
} // MicrowaveUI
```



RIT Department of Computer Science

Graphics

- Graphics can be drawn using a class which extends `JPanel`
- Swing will call the `paintComponent` method to draw:

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

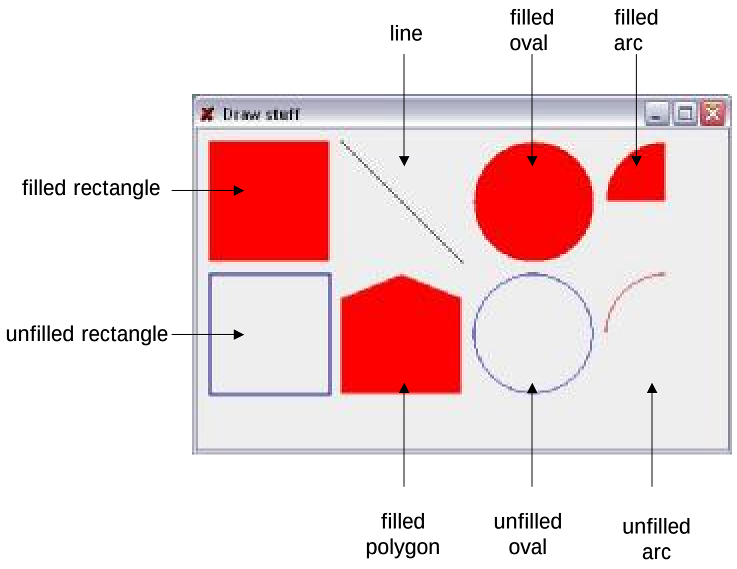
- There are a variety of drawing methods:

```
drawLine(int x1, int y1, int x2, int y2);
drawRect(int x, int y, int w, int h);
drawOval(int x, int y, int w, int h);
drawPolygon(int[] xpoints, int[] ypoints, int
npoints);
```

10/24/2005 Java GUI Programming 37

RIT Department of Computer Science

Graphics



10/24/2005 Java GUI Programming 38