

Basics of GUI

The background of the slide features a digital, futuristic landscape. In the foreground, a perspective grid of glowing purple lines recedes into the distance. In the background, a range of mountains is rendered in a wireframe style with glowing blue lines. The sky is a dark, deep blue with faint, wispy clouds.

GUI ?

File Look & Feel Themes

Table Demo Source Code

☒ Reordering allowed ☐ Column selection

☒ Horiz. Lines ☒ Row selection



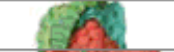







☒ Vert. Lines

Inter-cell spacing:

Row height:

Selection mode:

Autosize mode:

First Name	Last Name	Favorite Color	Favorite Movie	Favorite Number	Favorite Food
Mike	Albers	Green	Brazil	44	
Mark	Andrews	Blue	Curse of the Demon	3	
Brian	Beck	Black	The Blues Brothers	2.718	
Lara	Bunni	Red	Airplane (the whole series)	15	
Roger	Brinkley	Blue	The Man Who Knew Too Much	13	
Brent	Christian	Black	Blade Runner (Director's Cut)	23	
Mark	Davidson	Dark Green	Brazil	27	
Jeff	Dinkins	Blue	The Lady Vanishes	8	
Ewan	Dinkins	Yellow	A Bug's Life	2	
Amy	Fowler	Violet	Reservoir Dogs	3	



Support for GUI in Java

Support for GUI

- Abstract Windowing Toolkit (AWT) & Swing packages
 - Provides rich set of user interface components
 - `java.awt` & `javax.swing`
 - Old (AWT) VS. New (Swing)
- Components in awt & swing (start with J)
 - `Frame`, `JFrame`
 - `Menu`, `JMenu`
 - `Button`, `JButton`
 - `TextField`, `JTextFiled`
 - `Label`, `JLabel`
 - and many more....
- Use Java API Documentation well, its your FRIEND.

Abstract Windowing Toolkit

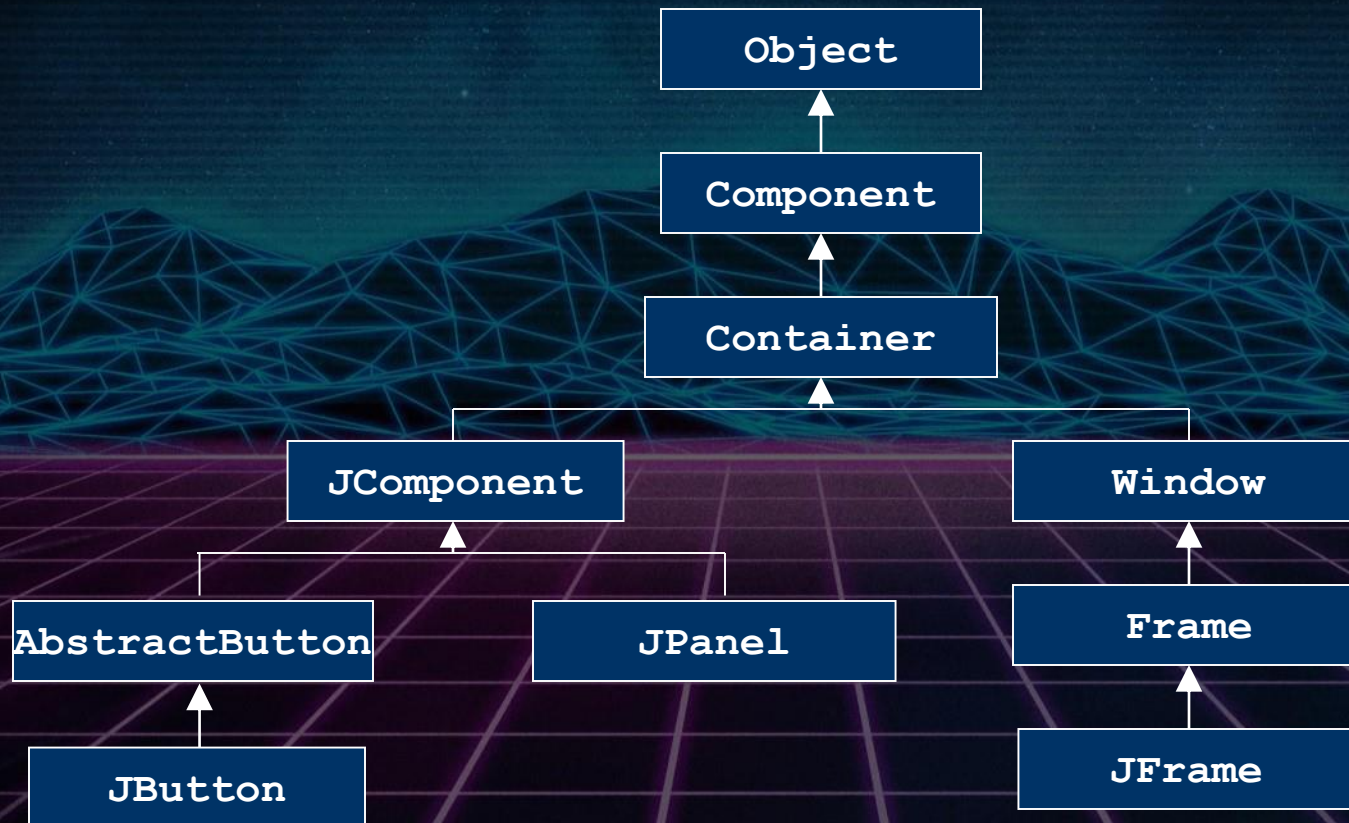
- AWT
 - The original GUI components
 - Referred as “Heavy Weight Components (HWC)”
 - Tied directly to the local platform’s GUI capabilities
 - Provides
 - robust event-handling model
 - Layout Managers

Swing

- Swing

- Newest GUI components, Names start with J can be identified
- “replacement” to the AWT
- Referred as “Light Weight Components (LWC)”
 - Swing components are written, manipulated and displayed completely in java
 - So they are not “weighed down” by the GUI capabilities of local platform
- Several Swing components are still HWC like JFrame etc.
- Allows uniform “look & feel” across all platforms

A Part of the Framework



GUI Creation Steps

The background of the slide is a digital landscape. The foreground is a dark blue floor with a glowing purple grid pattern that recedes into the distance. In the background, there are three mountain-like shapes constructed from a network of glowing blue lines, creating a wireframe effect. The sky is a dark, deep blue with some faint, wispy clouds.

GUI Creation Steps

1. import required package

- e.g. swing, awt

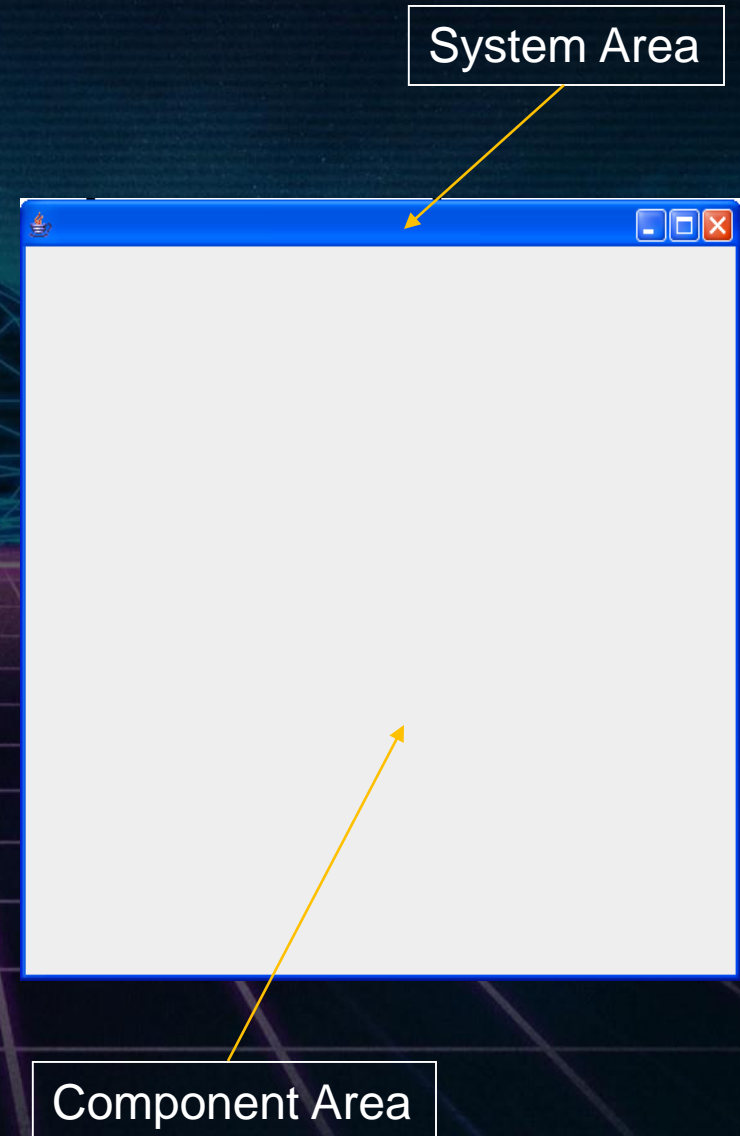
2. Setup the top level container

- e.g. `JFrame myframe = new JFrame();`

GUI Creation Steps (cont.)

3. Get the component Area of the top level Container

```
Container c = myFrame.getContentPane();
```



GUI Creation Steps (cont.)

4. Apply layout to that Area

- `c.setLayout(new FlowLayout());`

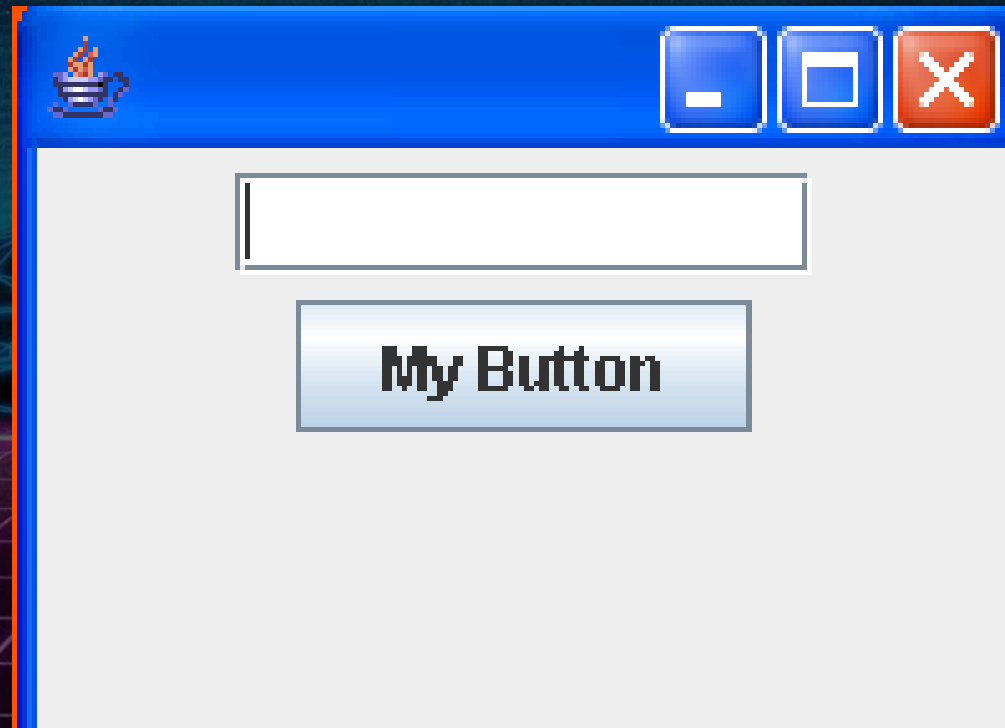
5. Create & add components

- `JButton b1 = new JButton("Hello");`
- `c.add(b1);`

6. Set size of Frame and make it Visible

- `myFrame.setSize(200,200);`
- `myFrame.setVisible(true);`

Example GUI



GUI: Example Code

//Step 1: import packages

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

```
public class GUITest {  
    JFrame myFrame ;  
    JTextField tf;  
    JButton b1;
```

```
public void initGUI ( ) { //method used for setting layout of GUI
```

//Step 2: setup the top level container

```
    myFrame = new JFrame();
```

//Step 3: Get the component area of top-level container

```
    Container c = myFrame.getContentPane();
```

//Step 4: Apply layouts

```
    c.setLayout( new FlowLayout( ) );
```

```
    ....
```

GUI: Example Code (cont.)

//Step 5: create & add components

JTextField tf = new JTextField(10);

JButton b1 = new JButton("My Button");

c.add(tf);

c.add(b1);

//Step 6: set size of frame and make it visible

myFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

myFrame.setSize(200,150);

myFrame.setVisible(true);

} //end init method

public GUITest () { // constructor

initGUI ();

}

.....

GUI: Example Code (cont.)

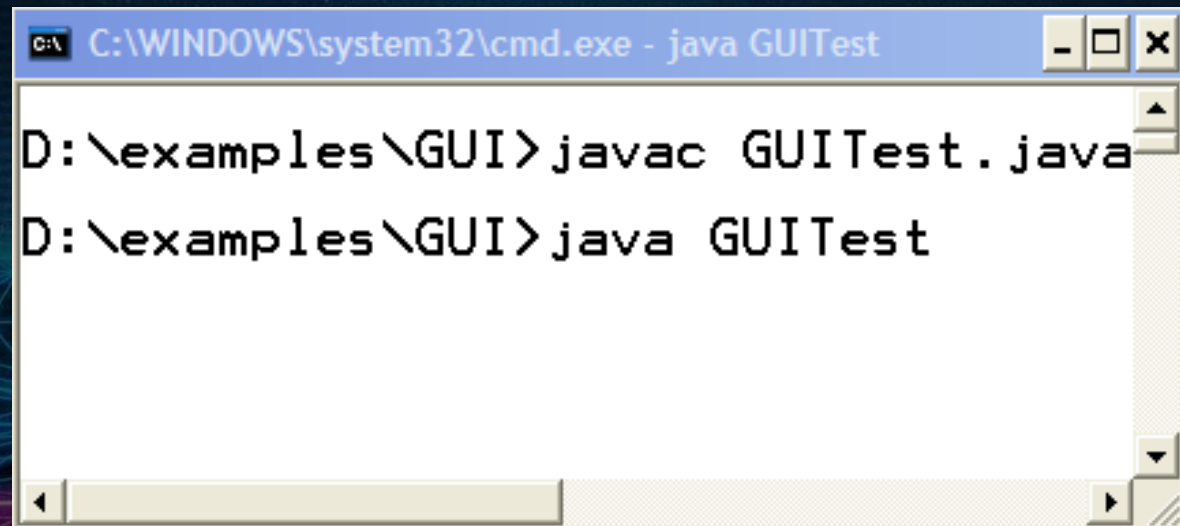
```
public static void main (String args[ ]) {
```

```
    GUITest gui = new GUITest();
```

```
}
```

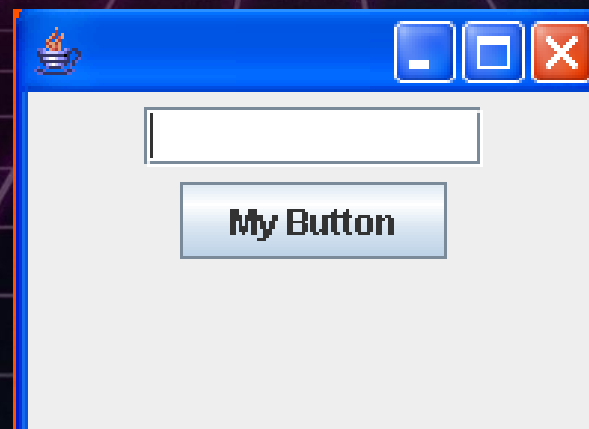
```
} // end of class
```

Compile & Execute



```
C:\WINDOWS\system32\cmd.exe - java GUITest

D:\examples\GUI>javac GUITest.java
D:\examples\GUI>java GUITest
```



GUI Creation Approaches

Composition

```
class GUITest{  
    JFrame frame;  
    Container c;  
    public GUITest ( ) {  
        frame = new JFrame ( );  
        c = frame.getContentPane();  
        .....  
        frame.setVisible(true);  
    }  
    .....  
}
```

Inheritance

```
class GUITest extends JFrame{  
  
    Container c;  
    public GUITest ( ) {  
  
        c = getContentPane();  
        .....  
        setVisible(true);  
    }  
    .....  
}
```

Layout Managers

The background of the slide is a digital landscape. The foreground is a dark blue floor with a glowing purple grid pattern that recedes into the distance. In the background, there are three mountains rendered in a wireframe style with glowing blue lines. The sky is a dark, deep blue with some faint, wispy clouds.

Layout Managers

- The layout of components in a container is usually governed by layout managers
- Similar to HTML – policy, not position
 - Do not set explicit pixel sizes or positions of things
 - Layout Managers know the intent (policy)
 - Layout Managers apply the intent to figure out the correct size on the fly

Layout Managers

- Layout Managers
 - Java supplies many layout managers. Five commonly used are:
 - FlowLayout
 - GridLayout
 - BorderLayout
 - BoxLayout
 - GridBagLayout

Layout Managers

- Layout Managers

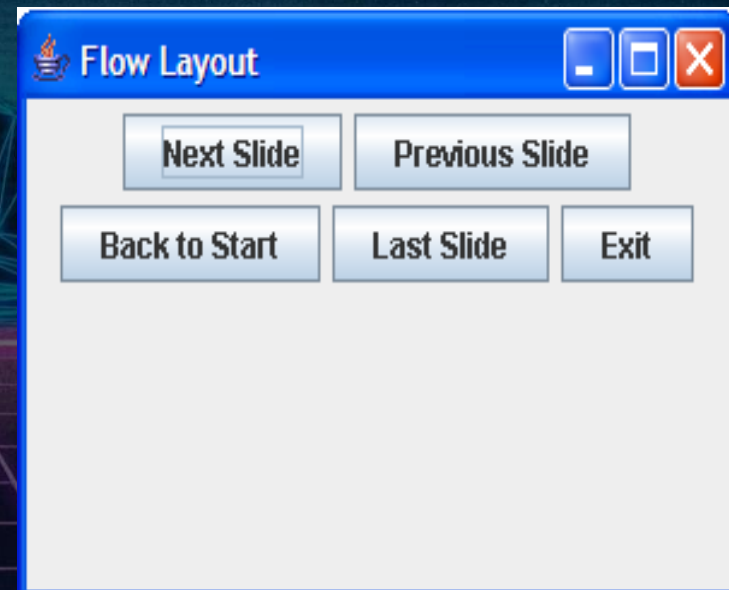
- FlowLayout

- Places components in a line as long as they fit, then starts the next line.
 - Uses “best judgement” in spacing components. Centers by default.
 - Lets each component assume its natural (preferred) size.
 - Often used for placing buttons on panels.

GUI: Example Code

FlowLayout

```
....  
c.setLayout (new FlowLayout() );  
  
JButton b1 = new JButton("Next Slide");  
JButton b2 = new JButton("Previous Slide");  
JButton b3 = new JButton("Back to Start");  
JButton b4 = new JButton("Last Slide");  
JButton b5 = new JButton("Exit");  
  
c.add(b1);  
c.add(b2);  
c.add(b3);  
c.add(b4);  
c.add(b5);  
}  
//end of main  
}
```



Layout Managers

- Layout Managers
 - GridLayout
 - Splits the panel into a grid with given number of rows and columns.
 - Places components into the grid cells.
 - Forces the size of each component to occupy the whole cell.
 - Allows additional spacing between cells.

GUI: Example Code

GridLayout

```
....  
c.setLayout (new GridLayout(3 , 2) );  
  
JButton b1 = new JButton("Next Slide");  
JButton b2 = new JButton("Previous Slide");  
JButton b3 = new JButton("Back to Start");  
JButton b4 = new JButton("Last Slide");  
JButton b5 = new JButton("Exit");  
  
c.add(b1);  
c.add(b2);  
c.add(b3);  
c.add(b4);  
c.add(b5);  
  
.....  
}
```



GUI: Example Code

GridLayout

```
....  
c.setLayout (new GridLayout(3 , 2, 10, 20 ) );
```

```
JButton b1 = new JButton("Next Slide");  
JButton b2 = new JButton("Previous Slide");  
JButton b3 = new JButton("Back to Start");  
JButton b4 = new JButton("Last Slide");  
JButton b5 = new JButton("Exit");
```

```
c.add(b1);  
c.add(b2);  
c.add(b3);  
c.add(b4);  
c.add(b5);  
  
size(200, 200)
```

```
//end of main
```

```
}
```

Extra space
between the cells



Layout Managers

- Layout Managers
 - BorderLayout
 - Divides the area into five regions
 - Adds a component to the specified region
 - Forces the size of each component to occupy the whole region.



GUI: Example Code

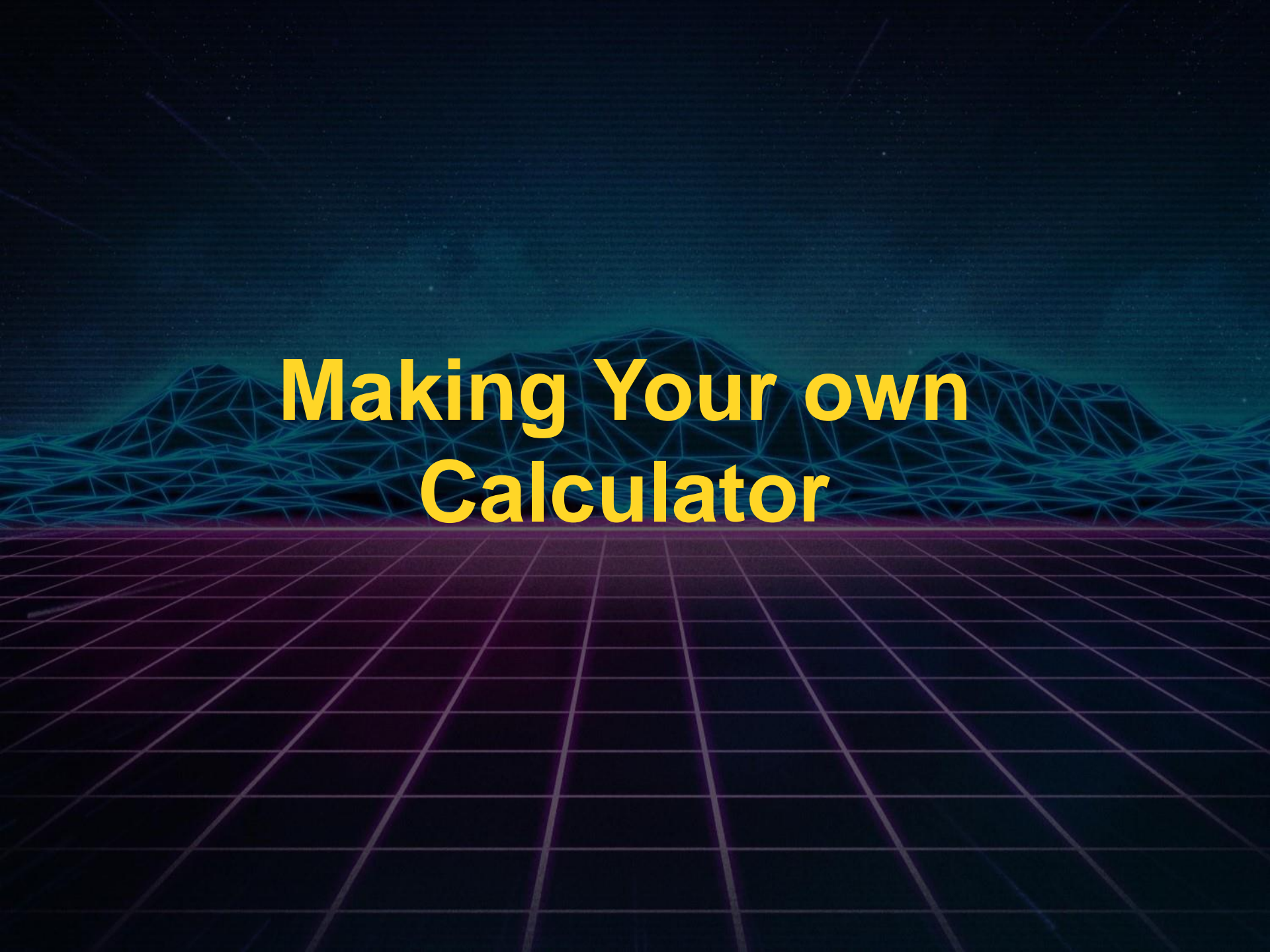
BorderLayout

```
....  
c.setLayout (new BorderLayout( ) );  
  
JButton b1 = new JButton("Next Slide");  
JButton b2 = new JButton("Previous Slide");  
JButton b3 = new JButton("Back to Start");  
JButton b4 = new JButton("Last Slide");  
JButton b5 = new JButton("Exit");  
  
c.add(b1, BorderLayout.NORTH);  
c.add(b2, BorderLayout.SOUTH);  
c.add(b3, BorderLayout.EAST);  
c.add(b4, BorderLayout.WEST);  
c.add(b5, BorderLayout.CENTER);  
}  
}
```



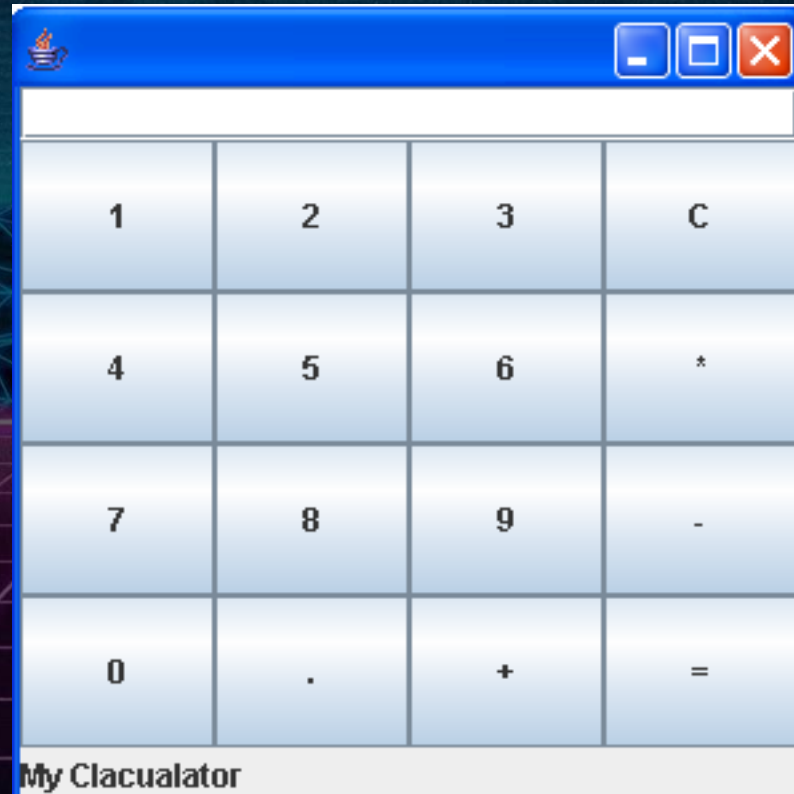
Layout Managers

- Layout Managers
 - Default Layouts
 - Each container has a default layout manager, which remains in effect until the component's `setLayout` method is called.
 - Some of the defaults are:
 - Content pane → BorderLayout
 - JPanel → FlowLayouts



Making Your own Calculator

Calculator GUI



Code: CalculatorGUI

```
import java.awt.*;  
import javax.swing.*;  
  
public class CalculatorGUI {  
  
    JFrame fCalc;  
  
    JButton b1, b2, b3, b4, b5, b6, b7, b8, b9, b0;  
    JButton bPlus, bMinus, bMul, bPoint, bEqual, bClear;  
  
    JPanel pButtons;  
  
    JTextField tfAnswer;  
  
    JLabel lMyCalc;
```

Code: CalculatorGUI (cont.)

```
public void initGUI ( ) { //used for setting layout of calculator
```

```
    fCalc = new JFrame();
```

```
    b0 = new JButton("0");
```

```
    b1 = new JButton("1");
```

```
    b2 = new JButton("2");
```

```
    b3 = new JButton("3");
```

```
    b4 = new JButton("4");
```

```
    b5 = new JButton("5");
```

```
    b6 = new JButton("6");
```

```
    b7 = new JButton("7");
```

```
    b8 = new JButton("8");
```

```
    b9 = new JButton("9");
```

```
    bPlus    = new JButton("+");
```

```
    bMinus   = new JButton("-");
```

```
    bMul      = new JButton("*");
```

```
    bPoint    = new JButton(".");
```

```
    bEqual    = new JButton("=");
```

```
    bClear    = new JButton("C");
```

```
    // continue....
```


Code: CalculatorGUI (cont.)

```
tfAnswer = new JTextField();
```

```
lMyCalc = new JLabel("My Clacualator");
```

```
//creating panel object and setting its layout  
pButtons = new JPanel (new GridLayout(4,4));
```

```
//adding components (buttons) to panel
```

```
pButtons.add(b1);  
pButtons.add(b2);  
pButtons.add(b3);  
pButtons.add(bClear);
```

```
pButtons.add(b4);  
pButtons.add(b5);  
pButtons.add(b6);  
pButtons.add(bMul);
```

```
//continue
```

Code: CalculatorGUI (cont.)

```
pButtons.add(b7);  
pButtons.add(b8);  
pButtons.add(b9);  
pButtons.add(bMinus);
```

```
pButtons.add(b0);  
pButtons.add(bPoint);  
pButtons.add(bPlus);  
pButtons.add(bEqual);
```

```
Container con = fCalc.getContentPane();  
con.setLayout(new BorderLayout());
```

```
//adding components to container  
con.add(tfAnswer, BorderLayout.NORTH);  
con.add(IMyCalc, BorderLayout.SOUTH);  
con.add(pButtons, BorderLayout.CENTER);
```

```
fCalc.setSize(300, 300);  
fCalc.setVisible(true);
```

```
} // end of initGUI method
```

```
.....
```


Code: CalculatorGUI (cont.)

//default constructor

```
public CalculatorGUI ( ) {  
    initGUI();  
}
```

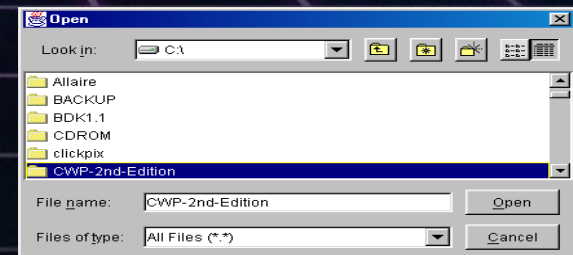
//main method

```
public static void main(String args[ ]) {  
  
    CalculatorGUI calGUI = new CalculatorGUI();  
}
```

```
} //end of class
```

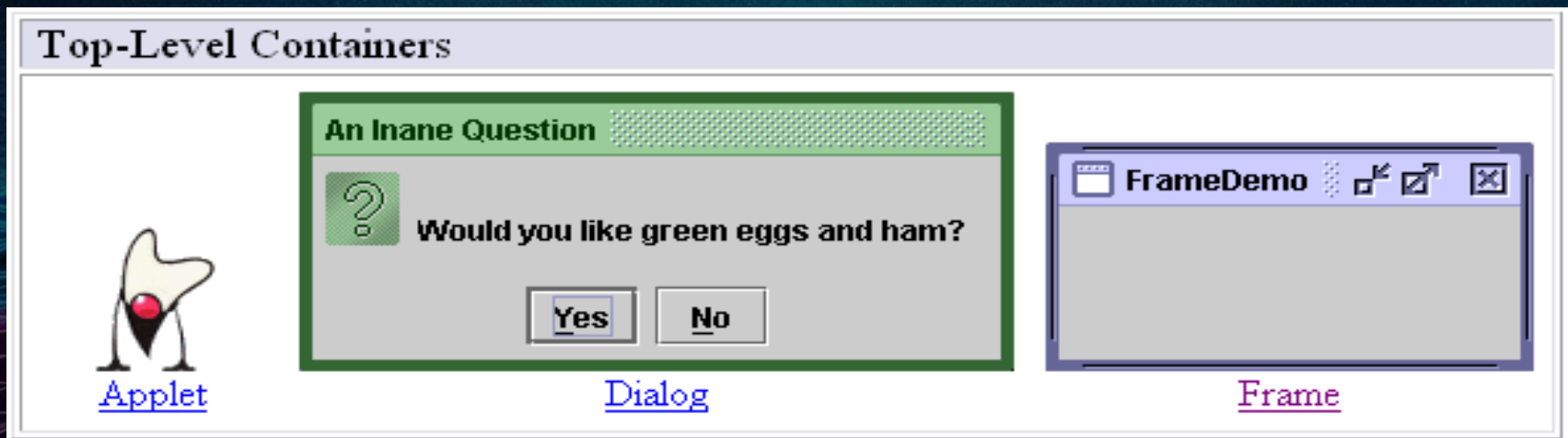
More Swing Components

- JCheckBox
 - Note uppercase B (vs. Checkbox in AWT)
- JRadioButton
 - Use a ButtonGroup to link radio buttons
- JTextArea
 - Place in JScrollPane if you want scrolling
- JFileChooser



Swing Components

Top Level Containers



Your application usually extends one of these classes !

Swing Components

- General Purpose Containers

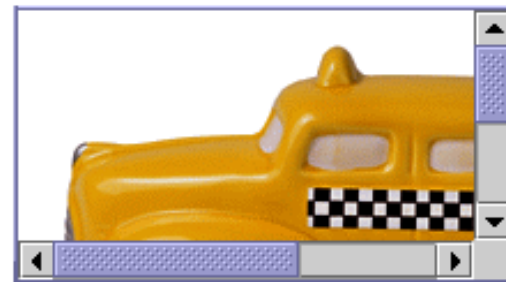
General-Purpose Containers

A Label on a Panel

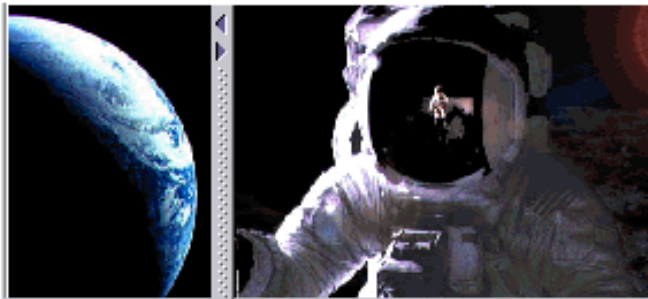
Color and font test:

- red
- blue
- green
- small

Panel



Scroll pane



Split pane



Tabbed pane



Tool bar