

# GUI Design

## Chapter 5 - Lecture Slides

(c) 2003-2008 E.S.Boese

1

## Containers

- Our applet is a container that we can add components to:

```
setLayout( new FlowLayout( ) );  
add( myLabel );  
add( myButton );
```

- We can also create panels to hold components with JPanel

```
JPanel pane;  
  
pane = new JPanel( );  
pane.setLayout( new FlowLayout( ) );  
pane.add( myLabel );  
pane.add( myButton );
```

(c) 2003-2008 E.S.Boese

2

## Containers

- Follow a layout manager
- A **Container** IS a **Component** (just like JButton and JTextField are Components), but it can also hold other containers and components.
  - e.g. JPanel
    - can put JButtons and JLabels in to a JPanel
    - can put other JPanels inside a JPanel
      - nest panels each with their own layout manager to fully control the layout
- No limit to number of Components a Container may hold
- Can put Containers within other Containers (JPanel in to the applet, JPanel in to another JPanel)
- Container has local coordinate system for Components within it (regardless of location on screen). Therefore, (0,0) is at the top left of the container

(c) 2003-2008 E.S.Boese

3

## Controlling the Layout

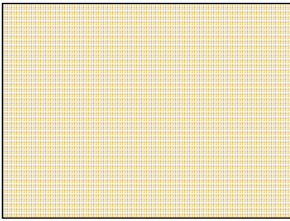
- Want to control *where* and *how* components are displayed
- *Layout managers* divide up the space of a container into regions and arrange components in these regions
- Some layout managers with adjust the sizes of components, other layout managers use the component's preferred size.

(c) 2003-2008 E.S.Boese

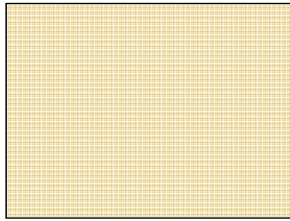
4

## Layout Examples

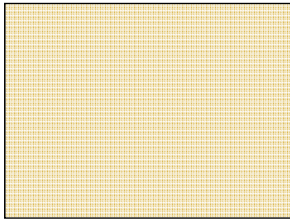
FlowLayout



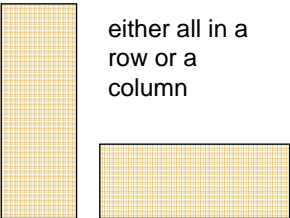
BorderLayout



GridLayout

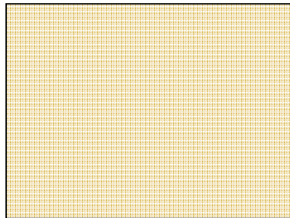


BoxLayout

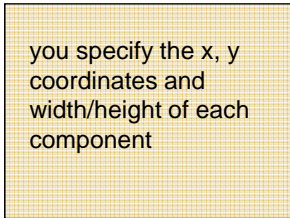


either all in a row or a column

GridBagLayout



null



you specify the x, y coordinates and width/height of each component

## Layout Managers

- We can use the `setLayout` method of a container to change its layout manager

on the applet:

```
setLayout( new BorderLayout( ) );
```

on a new panel:

```
JPanel panel;  
panel = new JPanel( );  
panel.setLayout(new BorderLayout());
```

## Add Components to container

- Add components to a container by calling the `add` method

```
void add( Component c )  
void add( Component c, int position )
```

- Example with FlowLayout:

```
add( component );
```

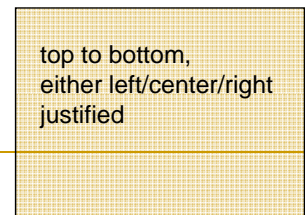
- Some layout managers require us to specify the position

e.g. BorderLayout:

```
add( component, BorderLayout.NORTH );
```

## FlowLayout

FlowLayout



## Flow Layout

- *Flow layout* puts as many components as possible on a row, then moves to the next row
- Rows are created as needed to accommodate all of the components
- Components are displayed in the order they are added to the container
- Each row of components is centered horizontally in the window by default, but could also be aligned left or right
- The horizontal and vertical gaps between the components can also be explicitly set

## FlowLayout

- Orders from
  - Left to right
  - By rows
  - From top to bottom
  - Centered by default
- Define alignment with constants
  - FlowLayout.LEFT
  - FlowLayout.CENTER
  - FlowLayout.RIGHT

## FlowLayout

- Create several ways:
  - default is horizontally centered

```
setLayout( new FlowLayout( ) );
```
  - set a particular alignment

```
setLayout( new FlowLayout( alignment ) );
```

where alignment is either:
    - FlowLayout.LEFT
    - FlowLayout.CENTER
    - FlowLayout.RIGHT
  - specify horizontal and vertical gaps between components – specified in pixels

```
setLayout( new FlowLayout( alignment, horizontalGap, verticalGap );
```

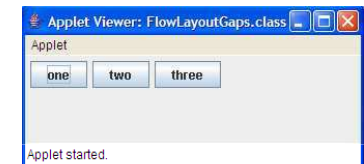
## FlowLayout

### Examples

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

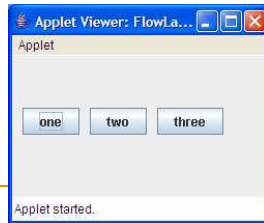
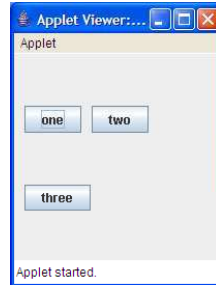
```
JPanel pa;  
pa = new JPanel( );  
pa.setLayout( new FlowLayout( FlowLayout.LEFT ) );
```

```
setLayout( new FlowLayout( FlowLayout.LEFT, 20, 10 ) );
```



## FlowLayout Example with gaps

```
import javax.swing.*;
import java.awt.*;
public class FlowLayoutGaps extends JApplet
{
    JButton one, two, three, four;
    public void init()
    {
        setLayout( new FlowLayout( FlowLayout.LEFT, 10, 50 ) );
        one = new JButton( "one" );
        two = new JButton( "two" );
        three= new JButton( "three" );
        add( one );
        add( two );
        add( three );
    }
}
```

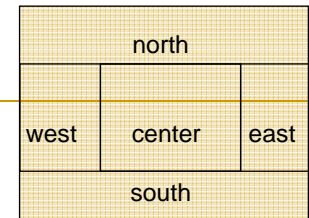


(c) 2003-2008 E.S.Boese

13

## BorderLayout

### BorderLayout

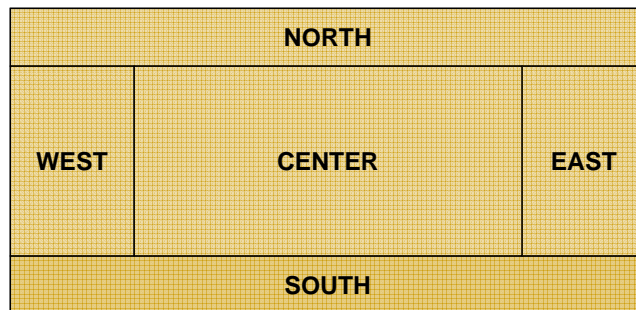


(c) 2003-2008 E.S.Boese

14

## BorderLayout

- *BorderLayout* has five (and only five) regions where components can be added:



(c) 2003-2008 E.S.Boese

15

## BorderLayout

- Container divided into 5 regions:
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
  - BorderLayout.EAST
  - BorderLayout.WEST
  - BorderLayout.CENTER

```
setLayout( new BorderLayout( ) );
add( myComponent , BorderLayout.NORTH);
add( myComponent2 , BorderLayout.CENTER);
```

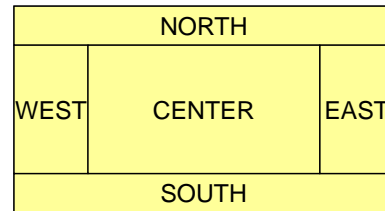
(c) 2003-2008 E.S.Boese

16

# BorderLayout

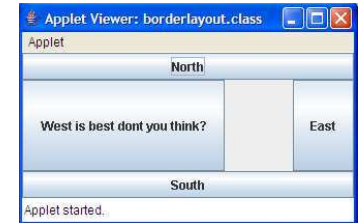
## ■ Rules to live by

- North and South
  - given preferred heights
  - Stretched horizontally to fill entire width
- East and West
  - Given preferred widths
  - Stretched vertically to fill space
- Center
  - Placed in remaining space
  - Stretched BOTH horizontally and vertically



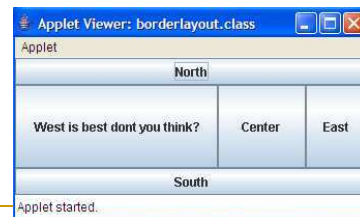
# BorderLayout Example

```
import java.awt.*;
import javax.swing.*;
public class BorderLayout extends JApplet
{
    JButton b1, b2, b3, b4, b5;
    public void init()
    {
        setLayout( new BorderLayout() );
        b1 = new JButton( "North" );
        b2 = new JButton( "South" );
        b3 = new JButton( "West is best dont you think?" );
        b4 = new JButton( "East" );
        b5 = new JButton( "Center" );
        add( b1, BorderLayout.NORTH );
        add( b2, BorderLayout.SOUTH );
        add( b3, BorderLayout.WEST );
        add( b4, BorderLayout.EAST );
    }
}
```



# BorderLayout2 Example

```
import java.awt.*;
import javax.swing.*;
public class BorderLayout2 extends JApplet
{
    JButton b1, b2, b3, b4, b5;
    public void init()
    {
        setLayout( new BorderLayout2() );
        b1 = new JButton( "North" );
        b2 = new JButton( "South" );
        b3 = new JButton( "West is best dont you think?" );
        b4 = new JButton( "East" );
        b5 = new JButton( "Center" );
        add( b1, BorderLayout2.NORTH );
        add( b2, BorderLayout2.SOUTH );
        add( b3, BorderLayout2.WEST );
        add( b4, BorderLayout2.EAST );
        add( b5, BorderLayout2.CENTER );
    }
}
```



# Structuring your code

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

public class Library extends JApplet
{
    // Declare all your variables HERE!
    // this includes all your widgets!

    public void init()
    {
        // set up a layout manager here
        doTitle();
        doLeftSide();
        doRightSide();
        doBottom();
    }

    public void doTitle()
    {
        // add title at the top
    }
}
```

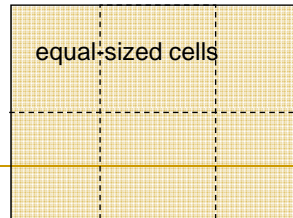
```
public void doLeftSide()
{
    JPanel leftside = new JPanel();
    // set the layout manager for the left side
    // set background color
    // add widgets
    // add this panel to the applet
}

public void doRightSide()
{
    JPanel rightside = new JPanel();
    // set the layout manager for the right side
    // set background color
    // add widgets
    // add this panel to the applet
}

public void doBottom()
{
    JPanel bottom = new JPanel();
    // set background color
    // add widgets
    // add this panel to the applet
}
}
```

# GridLayout

GridLayout



# GridLayout

- Similar to a spreadsheet
- Specify number of rows and columns
- As components are added, they're ordered top-to-bottom, left-to-right
- All cells of EQUAL size
  - all widths are the same, all heights the same
  - width not necessarily the same size as the height
- Stretches the component to the size of the cell

# GridLayout Example

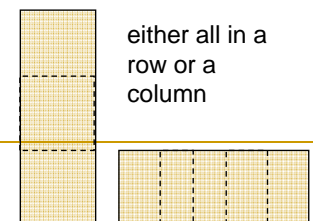
```
import java.awt.*;
import javax.swing.*;
public class GridSimple extends JApplet
{
    JButton b1, b2, b3, b4, b5;

    public void init()
    {
        setLayout( new GridLayout( 3, 2 ) );
        b1 = new JButton( "First" );
        b2 = new JButton( "Two" );
        b3 = new JButton( "Three" );
        b4 = new JButton( "Four" );
        b5 = new JButton( "Five" );
        add( b1 );
        add( b2 );
        add( b3 );
        add( b4 );
        add( b5 );
    }
}
```



# BoxLayout

BoxLayout

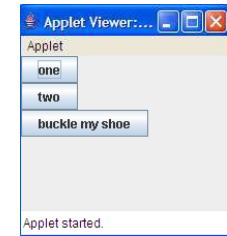
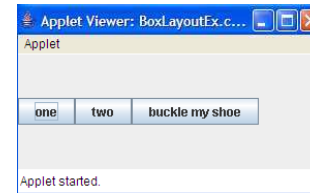


# BoxLayout

- Into a column or a row (you choose)
- Doesn't stretch components out like GridLayout
- Orientation:
  - BoxLayout.X\_AXIS
  - BoxLayout.Y\_AXIS

# BoxLayout - orientations

- BoxLayout.X\_AXIS
- BoxLayout.Y\_AXIS

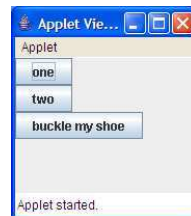


# BoxLayout - example

```
import javax.swing.*;

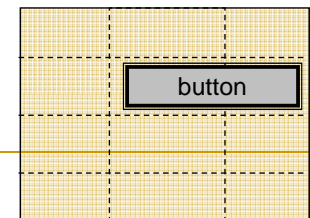
public class BoxLayoutEx extends JApplet
{
    JButton one, two, three;

    public void init( )
    {
        one = new JButton( "one" );
        two = new JButton( "two" );
        three = new JButton( "buckle my shoe" );
        setLayout( new BoxLayout( getContentPane( ), BoxLayout.Y_AXIS ) );
        add( one );
        add( two );
        add( three );
    }
}
```



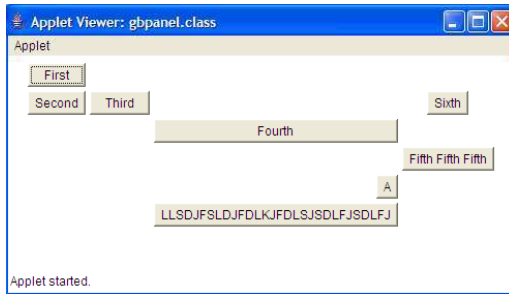
# GridBagLayout

GridBagLayout



## GridBagLayout

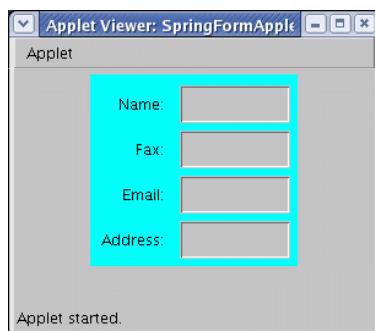
- Very flexible but complex to program
  - specify specific constraints



## SpringLayout

## SpringLayout

- Can use it similar to GridLayout, but allow components to be their own size (instead of stretching to fill up the cell)



## Warnings!

## Warning!

If you attempt to add a component that is already contained elsewhere, it will be removed and reinserted!!

Example:

```
JPanel panel;  
panel = new JPanel( );  
panel.setLayout( new BorderLayout( ) );  
panel.add( button1, BorderLayout.NORTH );  
panel.add( button2, BorderLayout.WEST );  
panel.add( button1, BorderLayout.EAST );
```

Then button1 will only be displayed in EAST!

## Warning!

- If you attempt to add a Component to a spot that is already filled and can only contain one component, the new component will only appear in the spot!!

Example:

```
JPanel panel = new JPanel( );  
panel.setLayout( new BorderLayout( ) );  
panel.add( button1, BorderLayout.NORTH );  
panel.add( button2, BorderLayout.NORTH );
```

Then only button2 will be displayed!

## Design Tip

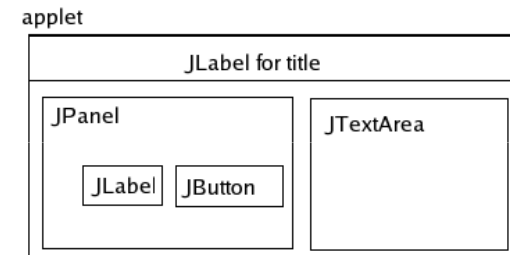
- If you don't want your Components to be resized during layout, you can place them in to a JPanel and add the JPanel to the Container

```
import java.awt.*;  
import javax.swing.*;  
  
public class GridSimple extends JApplet  
{  
    JButton b1, b2, b3, b4, b5;  
  
    public void init( )  
    {  
        setLayout( new GridLayout(3,2) );  
  
        b1 = new JButton( "First" );  
        b2 = new JButton( "Two" );  
        b3 = new JButton( "Three" );  
        b4 = new JButton( "Four" );  
        b5 = new JButton( "Five" );  
  
        add( b1 );    add( b2 );  
        add( b3 );    add( b4 );  
        add( b5 );  
    }  
}
```



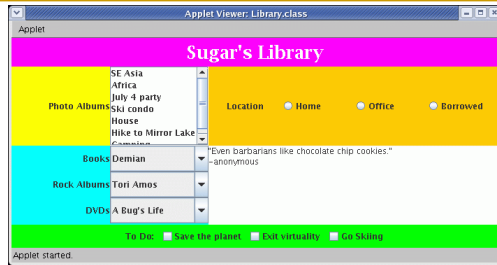
```
import java.awt.*;  
import javax.swing.*;  
public class GridSimplePanels extends JApplet  
{  
    JButton b1, b2, b3, b4, b5;  
    JPanel p1, p2;  
    public void init( )  
    {  
        setLayout( new GridLayout(3,2) );  
  
        b1 = new JButton( "First" );  
        b2 = new JButton( "Two" );  
        b3 = new JButton( "Three" );  
        b4 = new JButton( "Four" );  
        b5 = new JButton( "Five" );  
  
        p1 = new JPanel( new FlowLayout( ) );  
        p1.add(b1);  
        p1.add(b2);  
        p2 = new JPanel( new FlowLayout( ) );  
        p2.add(b3);  
        add( b3 );    add( b4 );    add( b5 );  
    }  
}
```

## Nesting JPanel in JApplet

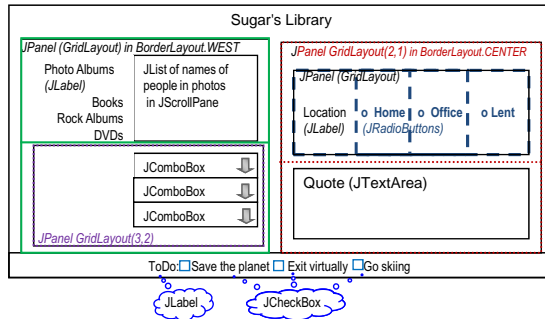


# Design

- Design



- Implementation



(c) 2005 by Elizabeth Sugar Boese

# SOME ideas in Designing a GUI

- **One Page** – Web pages are best if the user does not have to scroll.
- **Navigation:** Easily access different areas (e.g. JTabbedPane)
- **Recognition vs. recall** – allow users to select from a list (recognize) rather than have to enter the values (recall) e.g. states
- **Consistency** – keep site consistent – colors, location of buttons, toolbar, etc. Use same meaning for words: e.g. OK and Cancel

(c) 2005 by Elizabeth Sugar Boese

# SOME ideas in Designing a GUI

- **Progress Indicators** – keep users informed of what's going on, not just staring at an hourglass cursor!
- **Who are you?** – always add something about yourself to make it more human, and give a point of contact (email/phone/other)
- **Not too much stuff** – core features should be easily available (toolbar?), others less often used elsewhere (not in toolbar)
- **Text is readable** – not too small, not too big, and not too wide – 80 characters is the max a reader can handle

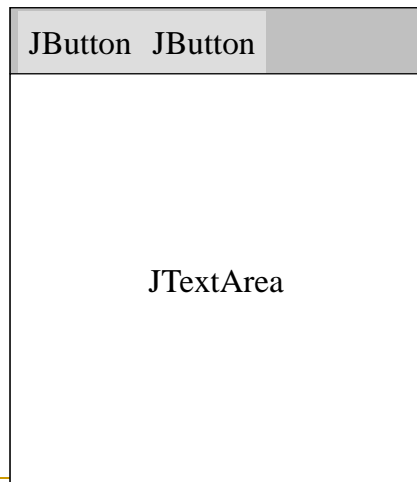
(c) 2005 by Elizabeth Sugar Boese

# Case Study



(c) 2003-2008 E.S.Boese

## Exercise: How would you get this?

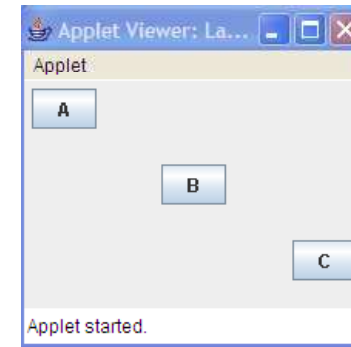


(c) 2003-2008 E.S.Boese

41

## Exercise: How would you get this?

Note: there are *many* different correct solutions to this problem!



(c) 2003-2008 E.S.Boese

42

## Summary

- Containers
- Layout managers
  - FlowLayout
  - BorderLayout
  - GridLayout
  - others
- Structuring your code
- Using JPanel
- Design

(c) 2003-2008 E.S.Boese

43