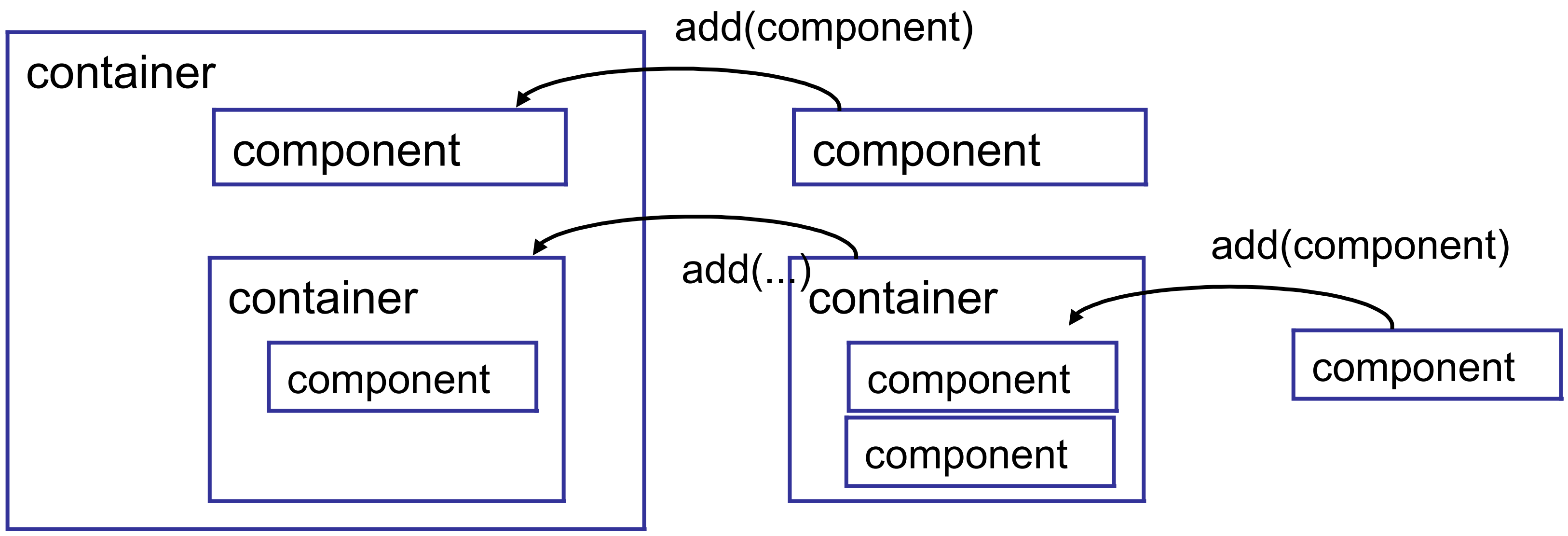
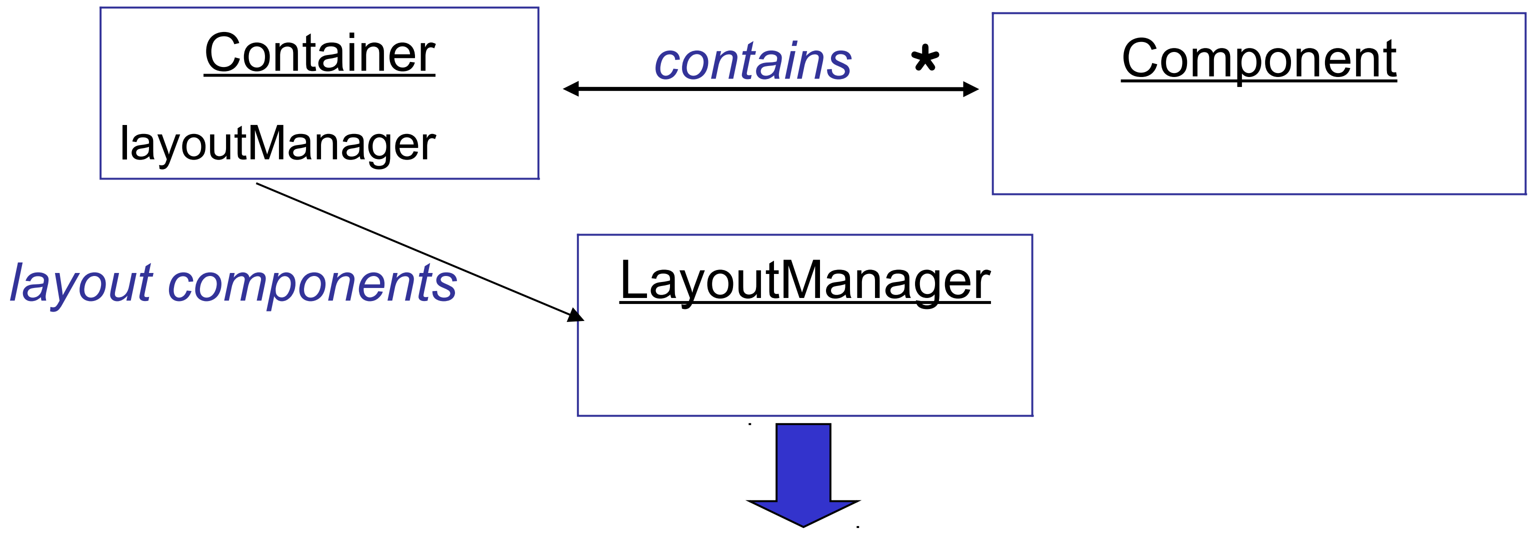
Containers and Components

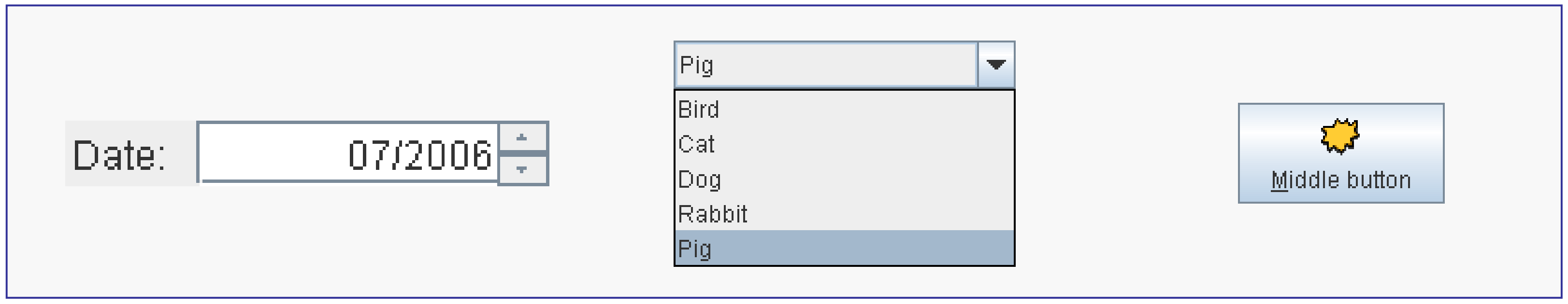
- A GUI has many **components** in **containers**.
- A container contains other components.
- A container is also a component; so a container may *contain other containers*.



Layout Managers

A container uses a **Layout Manager** to manage the position and size of components.





Why a layout manager?

Demo:

compare a Java application and Visual C# application when resizing a window.

In Java, the layout manager will rearrange or resize components.

In Visual C#, the components disappear.

Layout Managers

Classic Layout Managers are:

- BorderLayout (default for JFrame)
- FlowLayout (default for JPanel)
- BoxLayout
- GridLayout
- GridBagLayout
- CardLayout
- SpringLayout

add Layout Manager

- Use **`setLayout( )`** to assign a Layout Manager:

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

panel.setLayout( new GridLayout(3,4) );
```

Customizing a Layout

Some layout managers give you control over the layout.

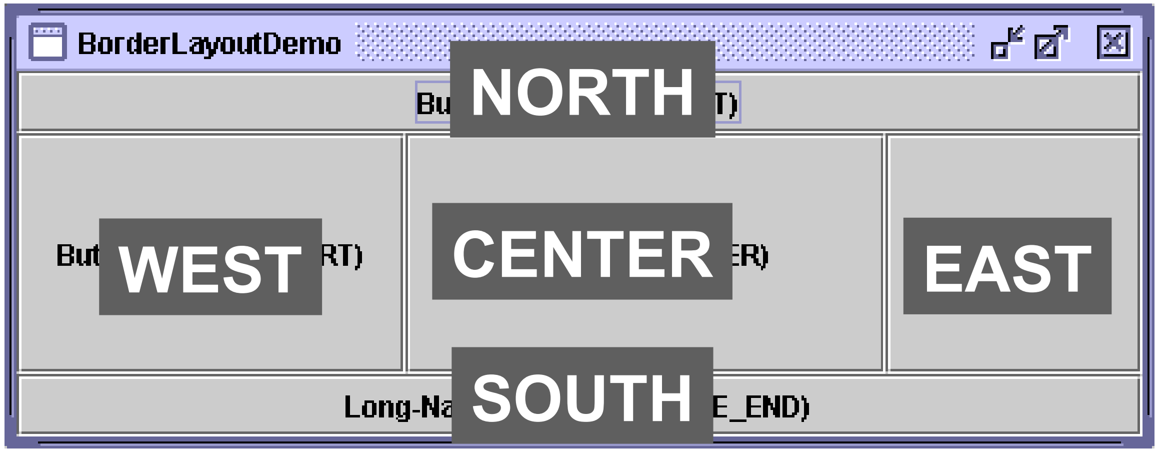
- FlowLayout, BorderLayout - set space between components

```
static final int GAP = 8;
FlowLayout layout = new FlowLayout( );
layout.setVGap( GAP );
layout.setHGap( GAP );
```

- GridBagLayout - almost *unlimited* control

BorderLayout

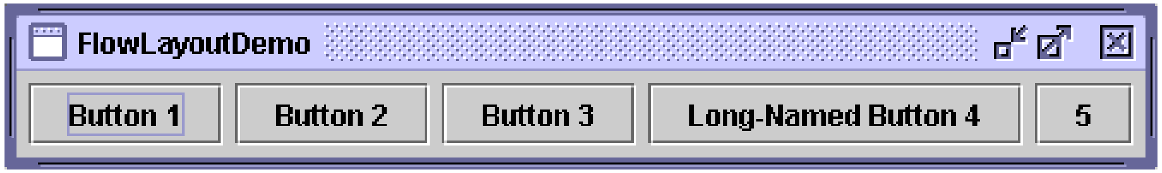
- BorderLayout divides the container into 5 zones.
- use: container.add(*component* , *WHERE*);
- If a zone is not used, other zones expand to use the space. CENTER gets the most space.



```
JButton button1 = new JButton("Button 1");
frame.add( button1, BorderLayout.EAST );
```

FlowLayout

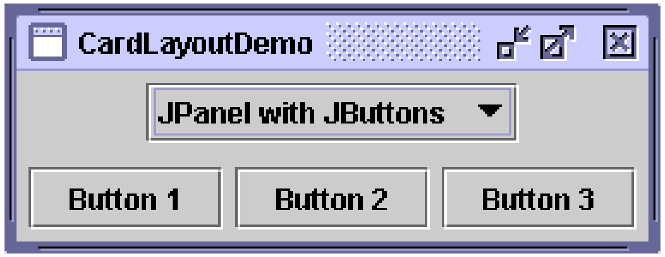
- "flows" the components into the available space.
- preserves the original (or requested) size of each component.
- components are added left to right, in order.

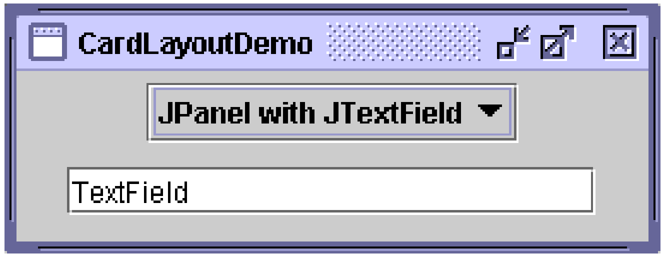


```
container.setLayout( new FlowLayout( ) );
JButton button1 = new JButton( "Button 1" );
JButton button2 = new JButton( "Button 2" );
container.add( button1 );
container.add( button2 );
```

CardLayout

- CardLayout lets you have different sets of components displayed at different times.
- One card is displayed at a time.
- Use next( ), first(), last() to change the displayed card.

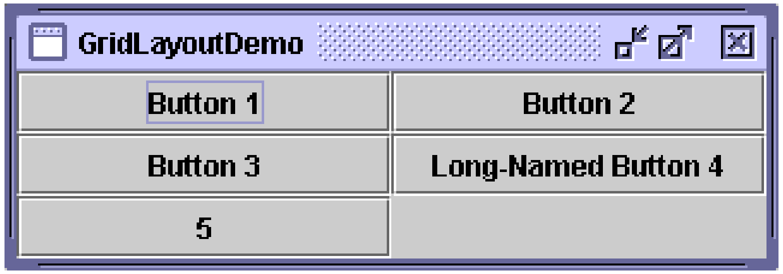




```
frame.setLayout( new CardLayout( ) );
// add buttons and panels to the cards
contentpane.add( button1 );
contentpane.add( button2 );
```

GridLayout

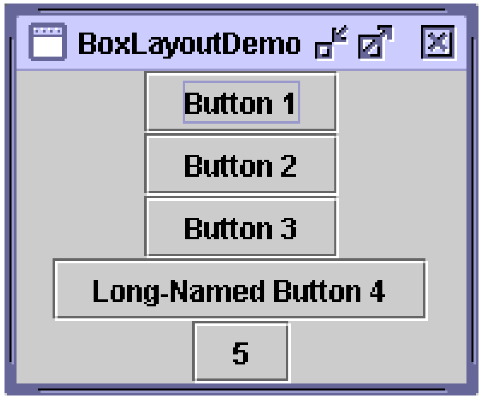
- specify a grid size (rows,cols) in constructor.
- Components are added to a grid, in the order that they are added.
- GridLayout makes all components have same size.



```
frame.setLayout( new GridLayout(3, 2) ); // (rows,cols)
// add buttons to the grid
frame.add( button1 );
frame.add( button2 );
frame.add( button3 );
```

BoxLayout

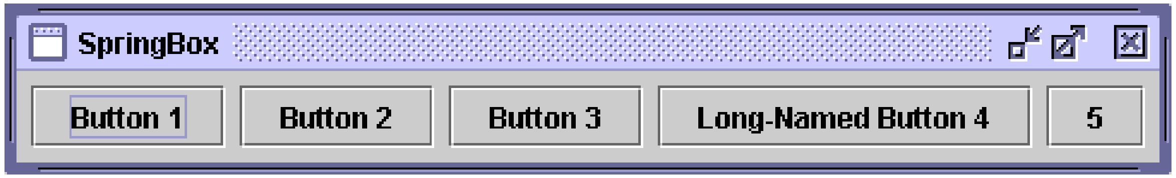
- BoxLayout puts components in a single row or column.
- It does not resize components.
- It allows different forms of component alignment.

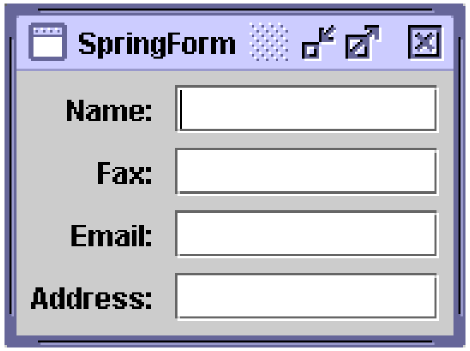


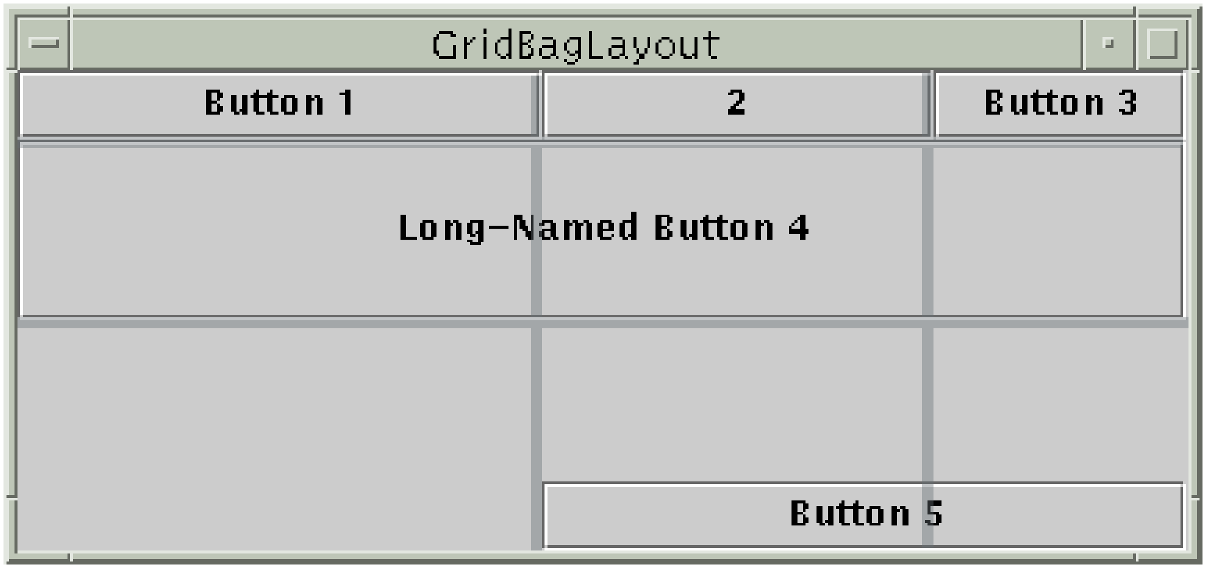
```
container.setLayout(    /* vertical box layout */
     new BoxLayout(container, BoxLayout.Y_AXIS) );
container.add( button1 );
container.add( button2 );
```

GridBagLayout and SpringLayout

- GridBagLayout and SpringLayout give you more control over the layout and sizing of components.
- can control margins, free space distribution, etc.







Newer Layout Managers

GroupLayout - treat several components as a group, so they can all have the same size or alignment. Makes layouts look much more professional.

FormLayout - layout a form containing labels and input areas in rows and columns, nicely aligned. It provides build-in data validators. Layout can be specified in text instead of Java code. FormLayout is a free, open-source component from http://www.jgoodies.com/freeware/libraries/forms/

AbsoluteLayout - put everything exactly where you say

(like in VisualStudio)

Have a JFrame or *Be* a JFrame?

Two styles of defining a UI class using JFrame:

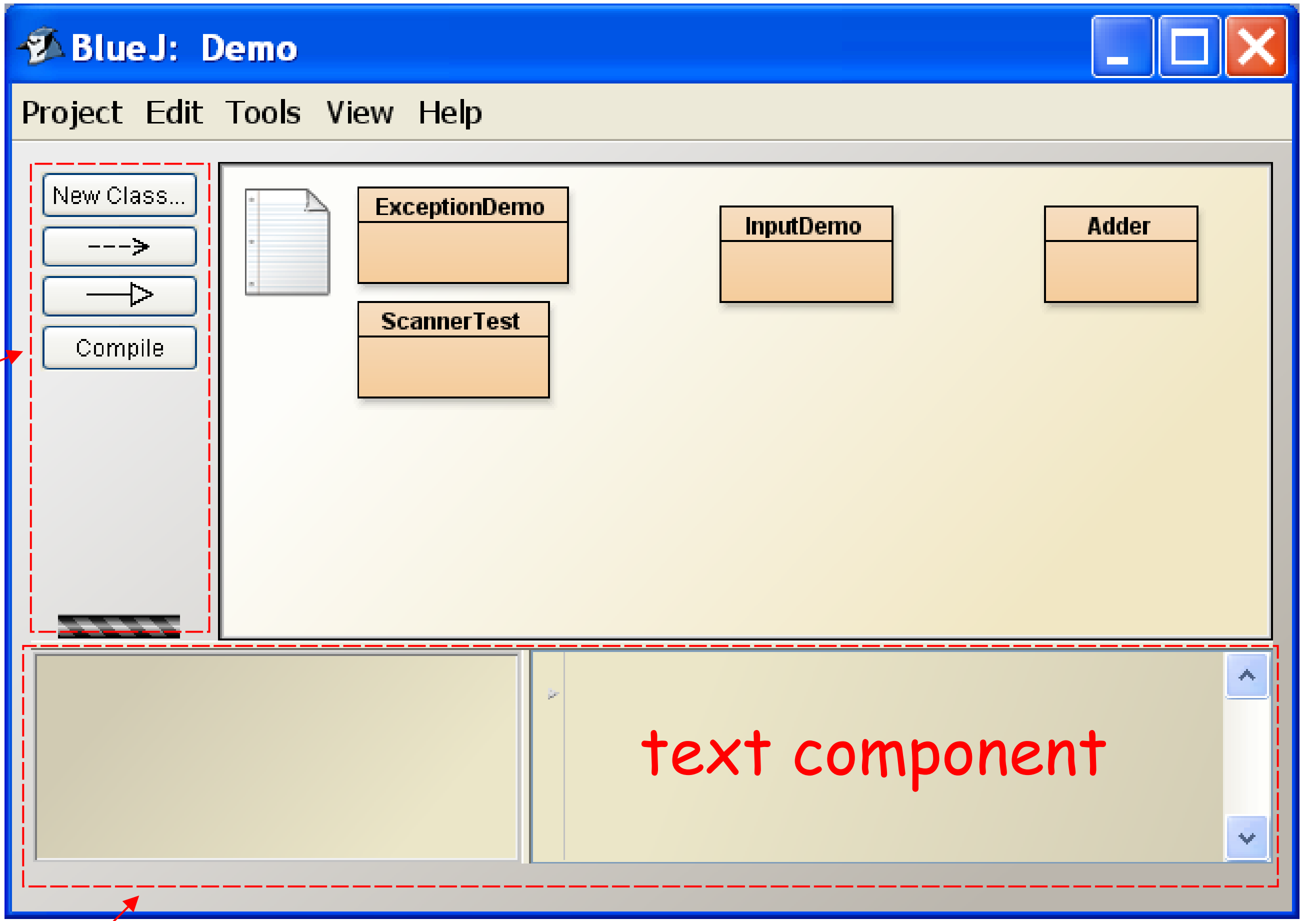
```
public class SwingExample {
   private JFrame frame;
   ...
   public SwingExample( ) {
      frame = new JFrame( );
     frame.setDefaultCloseOperation( ... );
```

```
public class SwingExample extemds JFrame {
   ...
   public SwingExample( ) {
      this.setDefaultCloseOperation( ... );
     initComponents( );
```

"`this`" *object is a JFrame.* *Don't create another one!!*

BlueJ: example of nested containers

Container with a row of buttons (buttons based on user prefs)



Container with 2 components inside

Lightweight Containers

A lightweight container is one that is not a window.

You must place it inside another container.

Cannot be drawn on screen by itself.

- JPanel simple rectangular area - most common
- JTabbedPane - multiple panels with a tab on top
- JSplitPane
- JInternalFrame - like a JFrame inside a JFrame

Nesting Containers

Example: a panel containing a text field and a button

```
JTextField textfield = new JTextField(12);
JButton button = new JButton("Login");
JPanel panel = new JPanel( );
panel.add( textfield );
panel.add( button );
```

Nesting: put the JPanel inside a JFrame

```
JFrame frame = new JFrame( );
frame.add( panel );
```

Benefit of LayoutManager

- What are the benefits of separating LayoutManager from the container classes?

  Why don't we put the layout code inside each container?