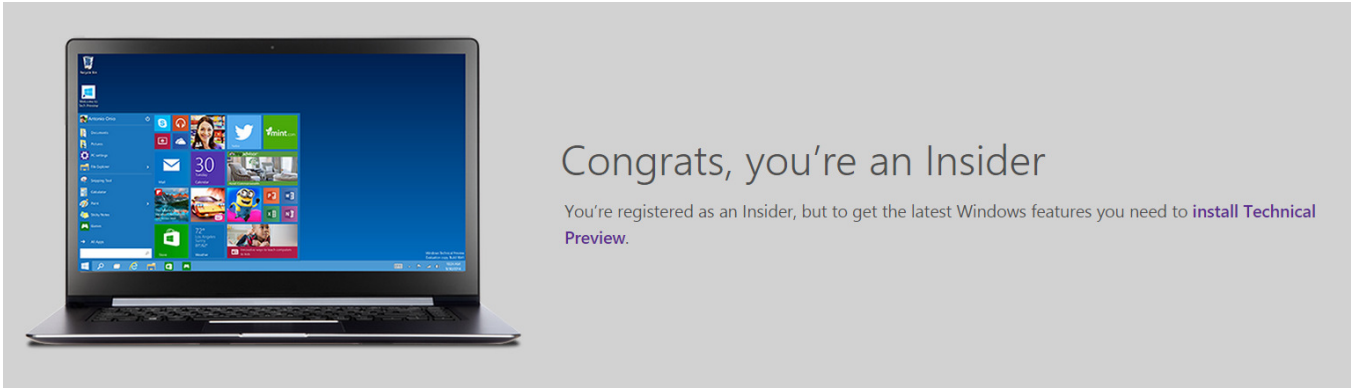
Windows 10 Technical Preview

- 1. Become an “Insider”



- 2. Check the requirements
- 3. Download the ISO image
- 4. Copy onto USB drive or DVD disc.
- 5. Install

Why “10”? And not “9”

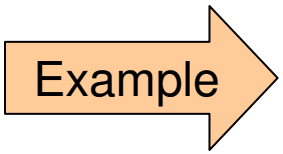
- Hypothesis 1
  - `if(version.StartsWith("Windows 9")) {`
  - `/* 95 and 98 */`
  - `}`
  - `else {`
- Hypothesis 2
  - Mac OS *X* (X is the Roman 10)
- Hypothesis 3
  - *You will get a free upgrade from “8” to “9”*

Laying Out Components

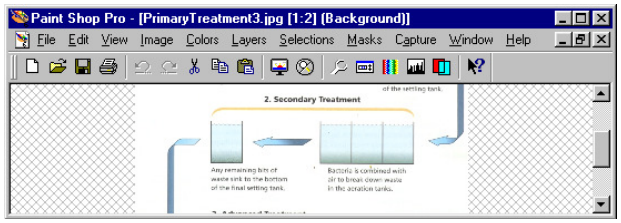
Interior Design for GUIs

What is Widget Layout?

- Positioning widgets in their container (typically a JPanel or a JFrame's content pane)
- Basic idea: each widget has a size and position
- Main problem: what if a window changes size?



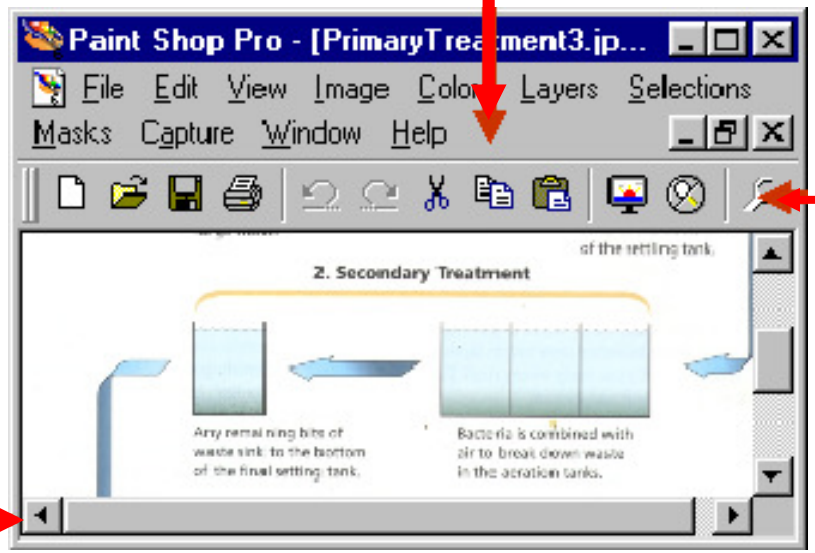
Resizing a Window



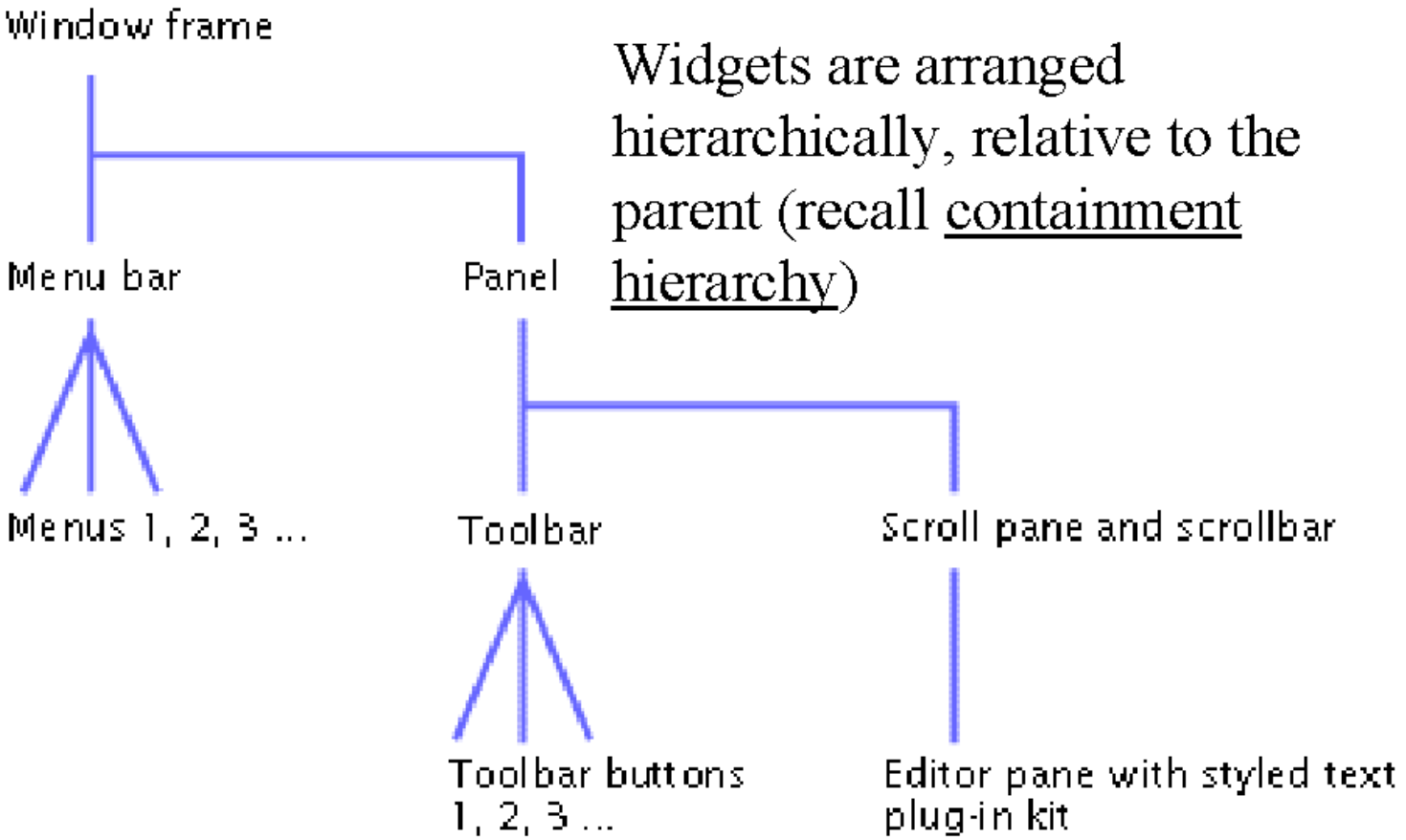
Menu wraps

Buttons lost

Scrollbar added

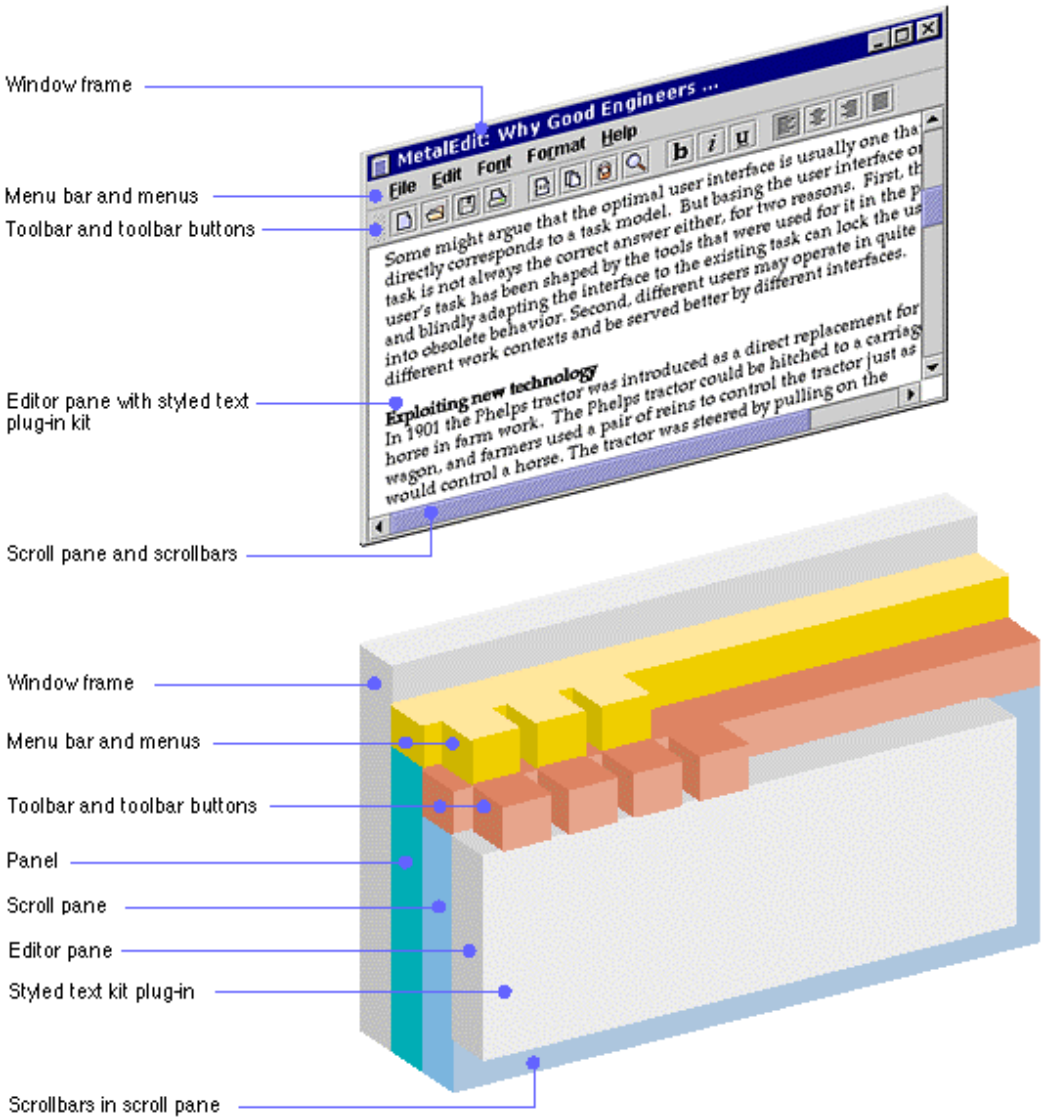


Hierarchical Widget Layout



Widgets are arranged hierarchically, relative to the parent (recall containment hierarchy)

Hierarchical Layout (2)



Size Properties

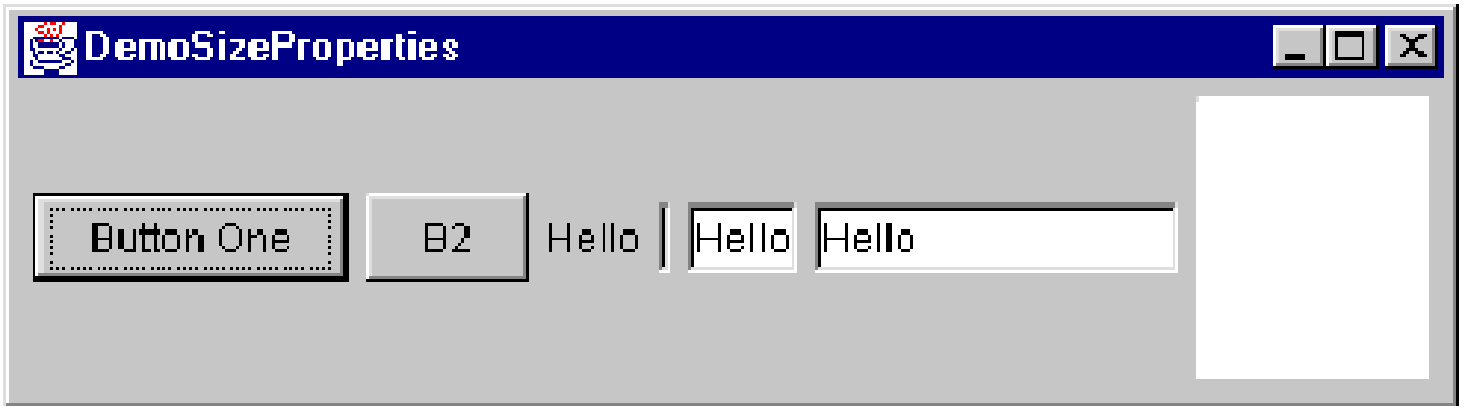
- When a component is instantiated, it takes on size properties…
  - Preferred size
  - Minimum size
  - Maximum size
- These properties are used to determine component size during (a) initial layout and (b) resize operations
- Size also affected by layout manager policies (more on this later)

Example Programs

DemoSizeProperties.java

DemoSize.java

DemoPosition.java



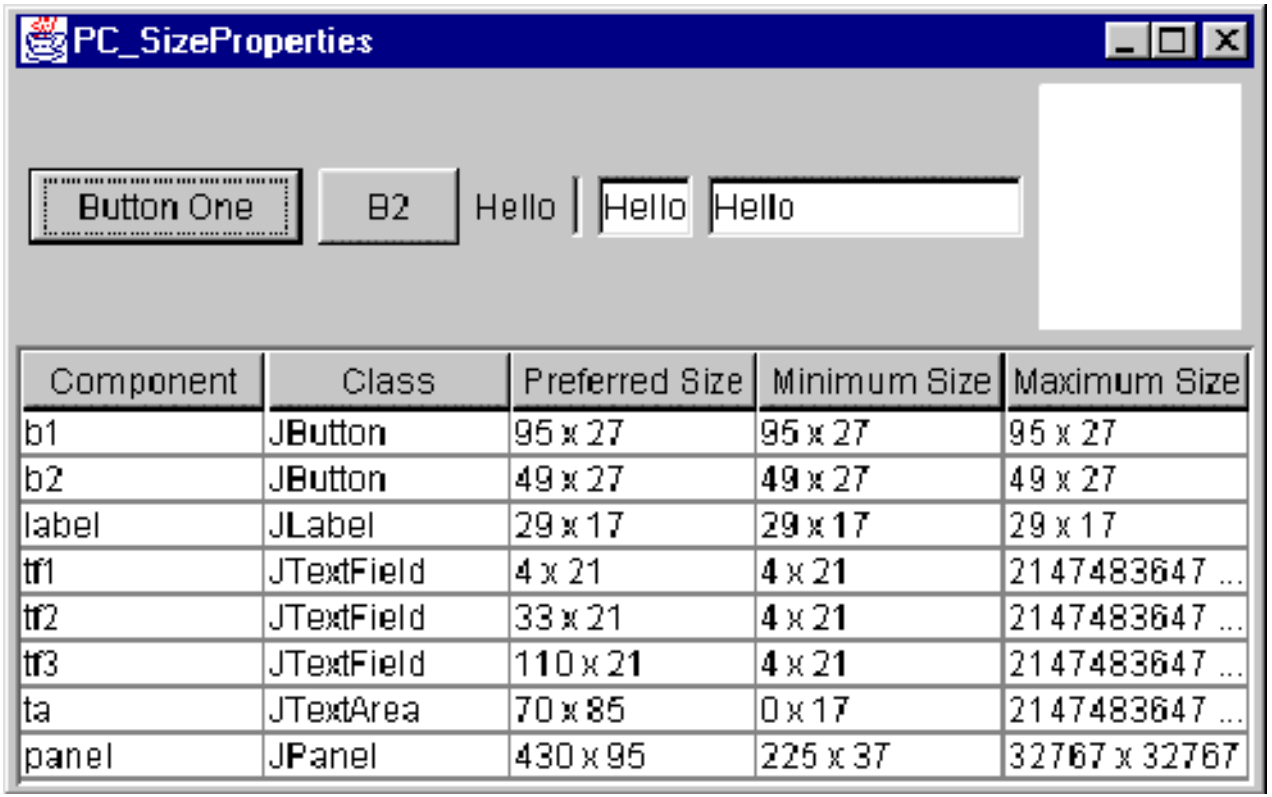
Progamming Challenge

- Create a new version of DemoSizeProperties that presents the component name, class, and size properties in a JTable
- Name the new program…

PC_SizeProperties.java

Solution

PC_SizeProperties.java



Component	Class	Preferred Size	Minimum Size	Maximum Size
b1	JButton	95 x 27	95 x 27	95 x 27
b2	JButton	49 x 27	49 x 27	49 x 27
label	JLabel	29 x 17	29 x 17	29 x 17
tf1	JTextField	4 x 21	4 x 21	2147483647 ...
tf2	JTextField	33 x 21	4 x 21	2147483647 ...
tf3	JTextField	110 x 21	4 x 21	2147483647 ...
ta	JTextArea	70 x 85	0 x 17	2147483647 ...
panel	JPanel	430 x 95	225 x 37	32767 x 32767

Widget Layout Models

- Absolute (aka fixed)
  - Control for component size and position
- Struts and springs
  - Control for component position
- Variable intrinsic size
  - Control for component size

Absolute Positioning

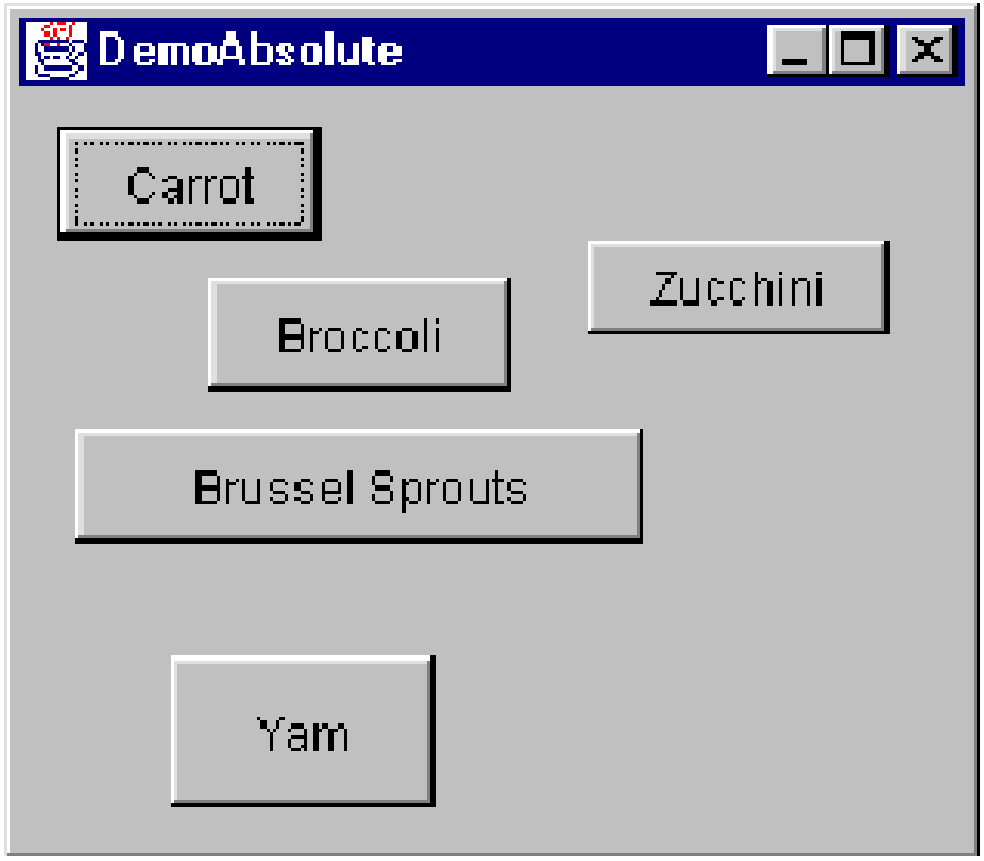
- Component position and size explicitly specified…
  - X and Y screen coordinates
  - Width and height of component
  - Units: pixels (typically)

Absolute Positioning (2)

- Advantages
  - Simple to implement
  - Widgets retain their position and size when window is resized (sometimes this is desireable)
  - Simple to build a resource editor (drag and drop widgets onto a screen; e.g., Visual Basic)
- Disadvantages
  - Difficult to change layout (too many ‘magic numbers’ or defined constants)
  - Poor response to resizing the window because…
    - Enlarging: too much empty space (ugly!)
    - Shrinking: components lost instead of wrapping

Example Program

DemoAbsolute.java

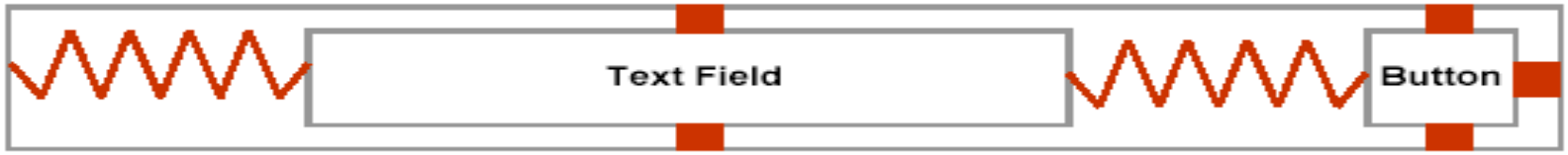


Struts and Springs

- Goals
  - Easy to use
  - Handles window resizing appropriately
- Idea
  - Add constraints to define geometric relationships between widgets

Struts and Springs (2)

- Place struts and springs into layout
- Struts ( ) - fixed regions (they don't change)
- Springs ( ) - can be compressed or stretched



- Advantage
  - When the window is resized, widget position is determined automatically by constraint equations

Variable Intrinsic Size

- Each component has intrinsic size properties (i.e., preferred size, minimum size, maximum size)
- During layout, components report their size properties to the layout manager (recursively, if necessary)
- Designers have limited control over this
  - Some layout managers respect size properties, while others do not!

Widget Communication

- Scenario #1: A scrollbar moves the enclosed text also moves
- Scenario #2: A window is resized components change in position and size
- How does this happen?
- Pseudo-events are used for widget to widget communication
- “Pseudo” because they are a responses to an implicit event (the reaction of one widget to another)
- Accomplished by subclassing and parent notification

Subclassing

- Most widgets are subclasses of other widgets
- Methods in superclass are overridden to handle updating the widget as well as notifying other widgets

Parent Notification

- Widgets notify their parent (enclosing widget) of any changes that have occurred
- Parent does the “right thing” (e.g., ask a scrollbar for its position and move the text window)

Java's Layout Managers

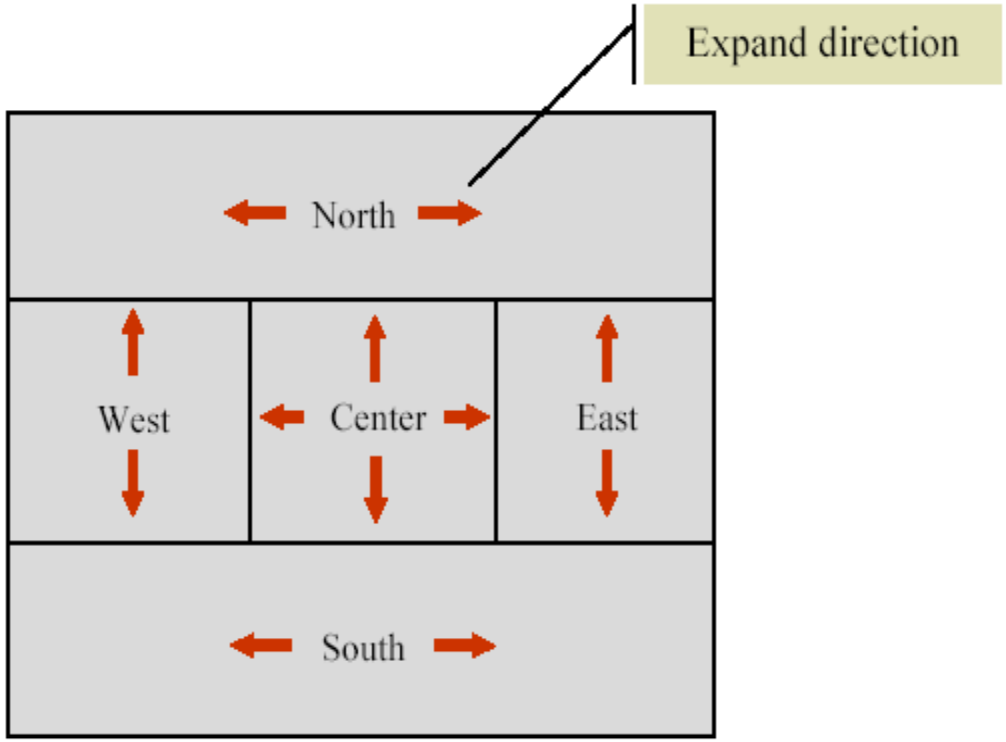
- BorderLayout
- FlowLayout
- GridLayout
- BoxLayout
- GridBagLayout
- CardLayout
- OverlayLayout
- etc.

BorderLayout

- Places components in one of five regions
  - North, South, East, West, Center
- Support for struts and springs
  - Struts (✓)
    - Can specify ‘hgap’, ‘vgap’
  - Springs (✗)
    - Inter-component space is fixed
- Support for variable intrinsic size (✓)
  - Components expand to fill space in region

Border Layout (2)

- Components ‘expand’ (or ‘stretch’) to fill space as follows

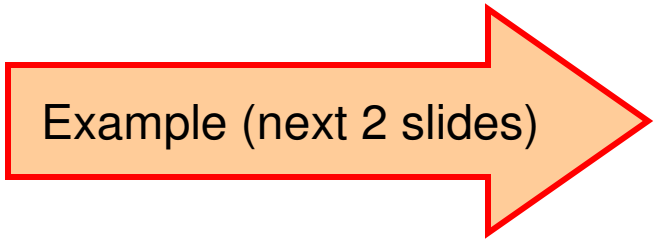


Example Program

DemoBorderLayout.java

usage: java DemoBorderLayout arg1

where 'arg1' = strut size in pixels

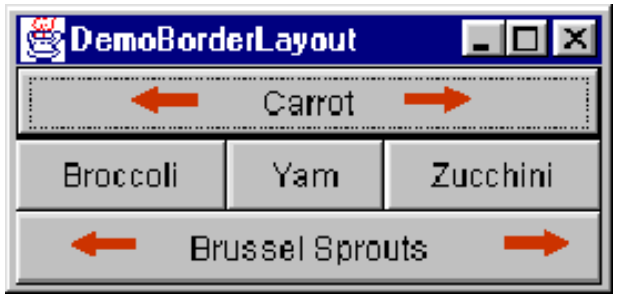


Example Program

No struts

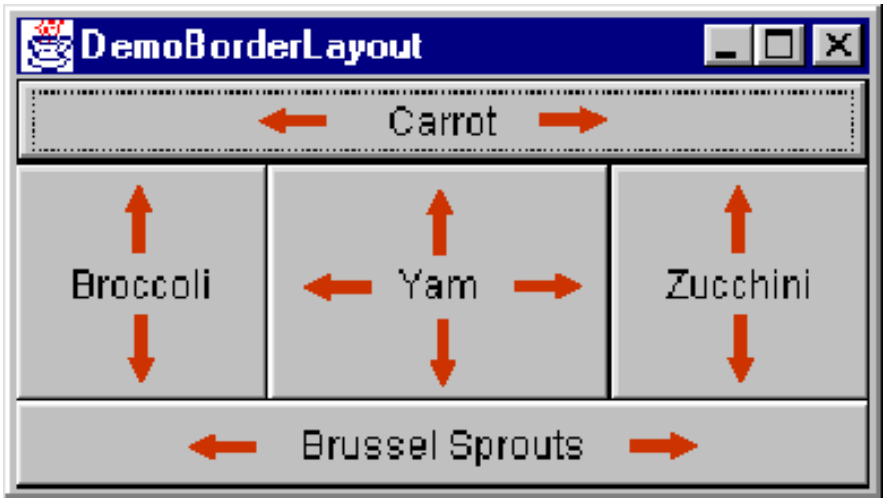
Invocation: java DemoBorderLayout 0

Launch



Variable intrinsic size

Resize

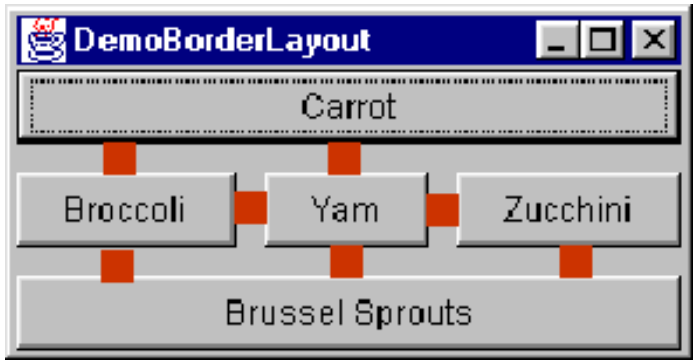


Example Program

With struts : hgap = vgap = 10 pixels

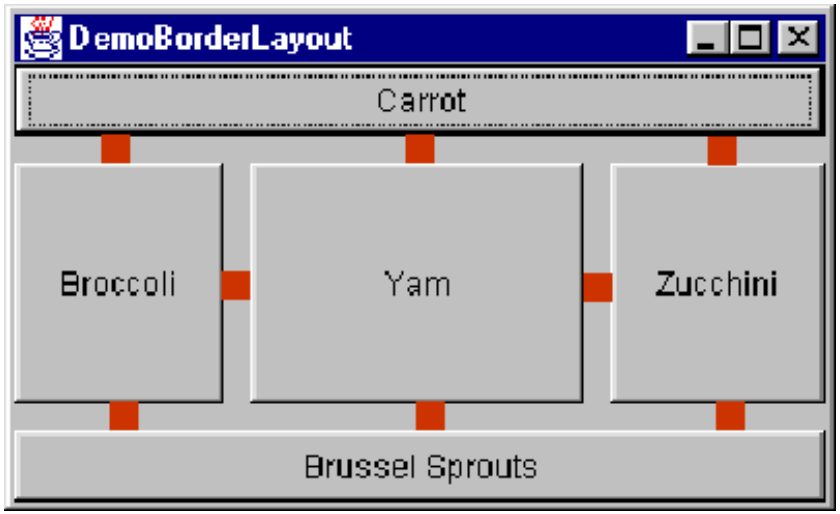
Invocation: java DemoBorderLayout 10

Launch



Struts

Resize



FlowLayout

- Arranges components in a group, left-to-right
- Group alignment: left, center, right
- Wraps components to new line if necessary
- Support for struts and springs
  - Struts (✓)
    - Can specify ‘hgap’, ‘vgap’
  - Springs (✗)
    - Inter-component space is fixed
- Support for variable intrinsic size (✗)
  - Component size is fixed
- Space is added before/after/below the entire group of components to fill available space

Example Program

DemoFlowLayout.java

usage: java DemoFlowLayout arg1 arg2

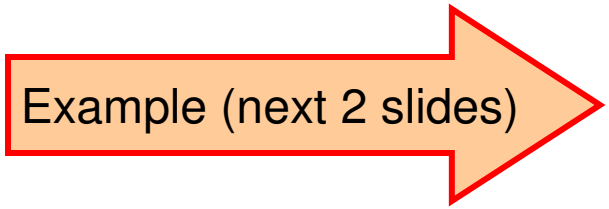
where 'arg1' = strut size in pixels

and 'arg2' is one of

c = center alignment

l = left alignment

r = right alignment



Example Program (2)

Example Program (2)
Default for FlowLayout…
struts : hgap = vgap = 5,
alignment = center

Invocation: java DemoFlowLayout 5 c

Launch

Resize

Fill available space before/after/below group of components

Example Program (3)

With struts : hgap = vgap = 10, alignment = right

Invocation: java DemoFlowLayout 10 r

Launch

Resize

GridLayout

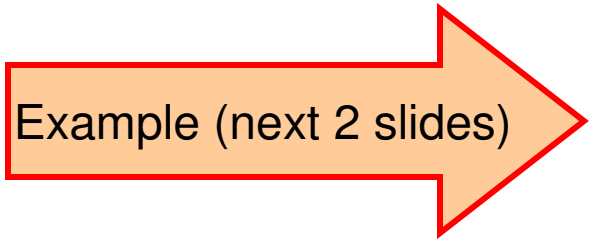
- Arranges components in a rectangular grid
- The grid contains equal-size rectangles
- Support for struts and springs
  - Struts (✓)
    - Can specify ‘hgap’, ‘vgap’
  - Springs (✗)
    - Inter-component space is fixed
- Support for variable intrinsic size (✓)
- Components expand to fill rectangle

Example Program

DemoGridLayout.java

usage: java DemoGridLayout arg1

where 'arg1' = strut size in pixels



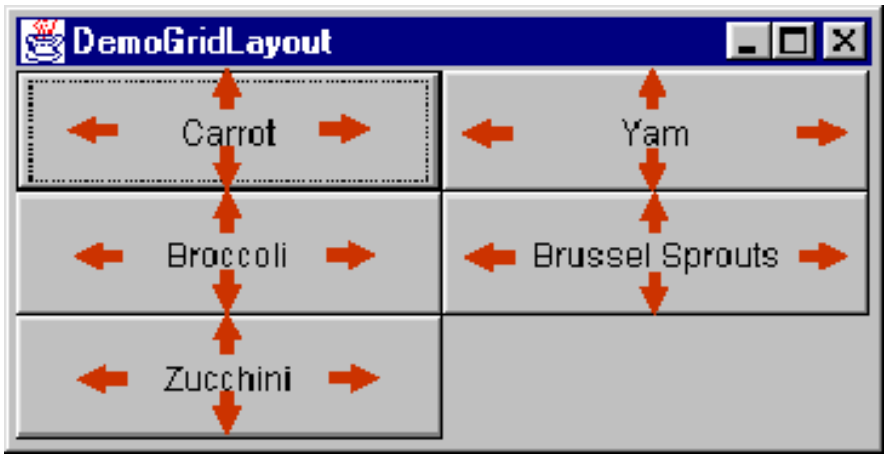
Example Program (2)

No struts

Invocation: java DemoGridLayout 0

Launch

Resize



Equal-size rectangles

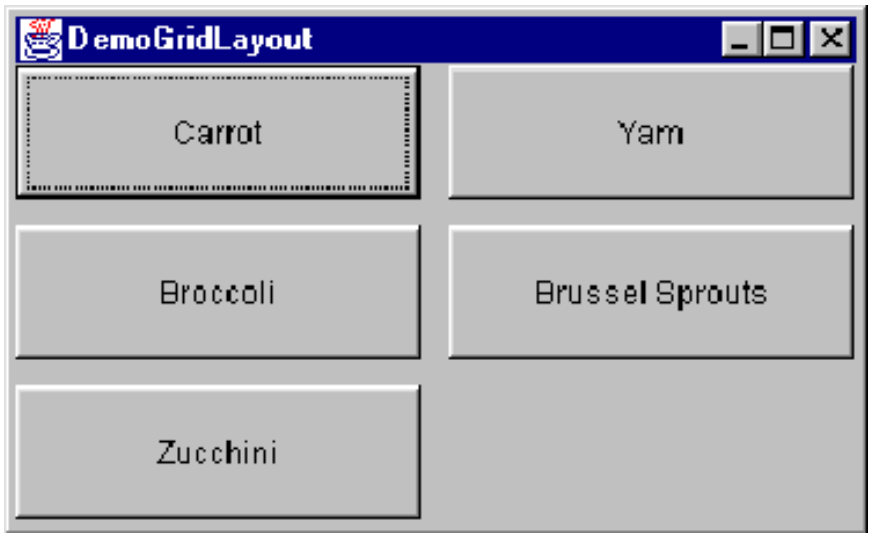
Example Program (3)

With struts : hgap = vgap = 10

Invocation: java DemoGridLayout 10

Launch

Resize



BoxLayout

- Arranges components vertically or horizontally
- Components do not wrap
- Support for struts and springs
  - Struts (✓)
    - Can specify ‘rigid areas’
  - Springs (✓)
    - Can specify ‘horizontal glue’ or ‘vertical glue’
- Support for variable intrinsic size (✓)
  - Components expand if maximum size property is set

Example Program

DemoBoxLayout.java

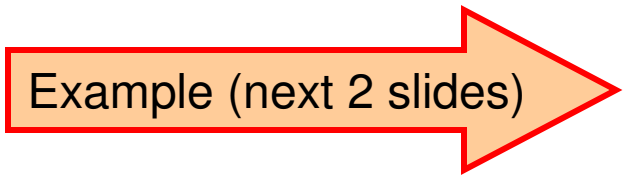
usage: java DemoBoxLayout arg1 arg2

where 'arg1' is one of

- c = centre alignment
- l = left alignment
- r = right alignment

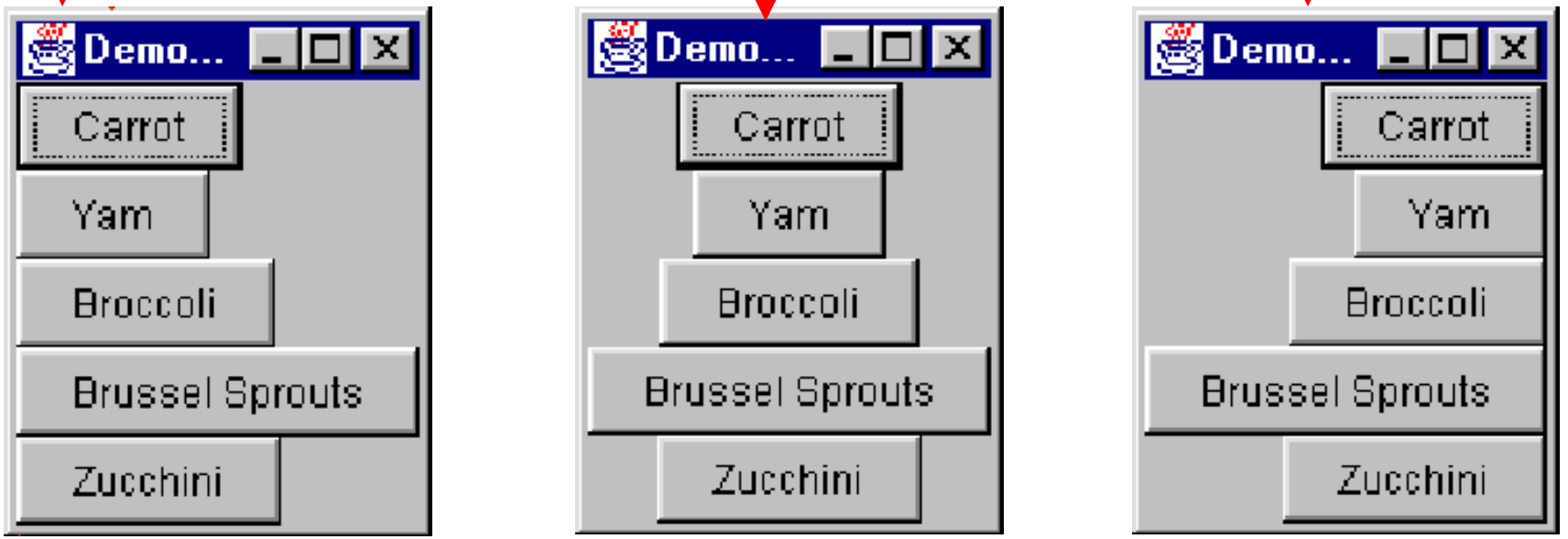
and 'arg2' is one of

- e = enable struts and springs demo
- d = disable struts and springs demo



Example Program (2)

Invocation: java DemoBoxLayout r d
Invocation: java DemoBoxLayout l d
Invocation: java DemoBoxLayout c d



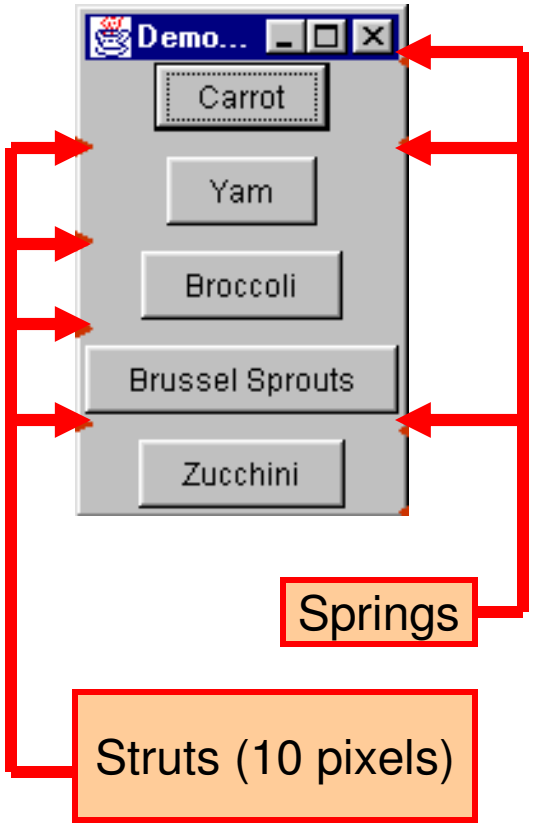
Default is left align

Example Program (3)

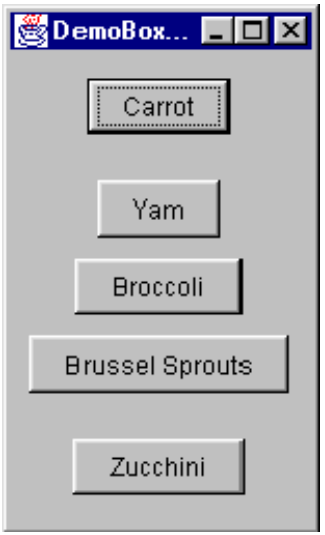
Enable struts and springs demo

Invocation: java DemoBoxLayout c e

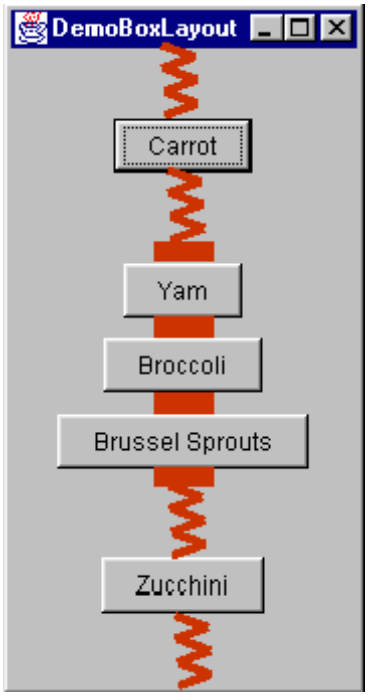
Launch



Resize



Resize more



Size Control

- How is a component's size determined during layout and during resize operations?
- Three factor determining component sizes:
  - The component's size properties (preferred, minimum, and maximum)
  - Whether the size properties are "assumed" or "explicitly set"
  - Layout manager policies

Example Program

DemoSizeControl.java

usage: java DemoSizeControl arg1

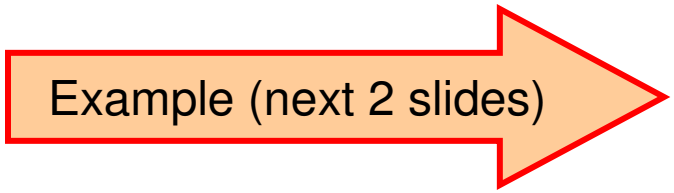
where 'arg1' specifies the layout manager as follows:

1 = BorderLayout

2 = FlowLayout

3 = GridLayout

4 = BoxLayout

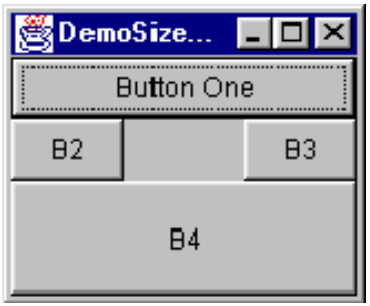


Example Program (2)

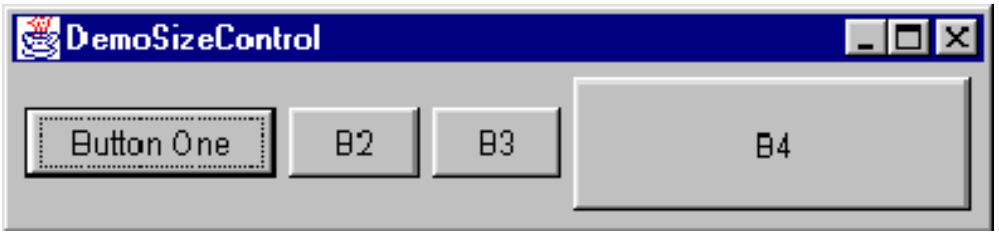
Component construction and configuration

```
JButton b1 = new JButton("Button One");
JButton b2 = new JButton("B2");
JButton b3 = new JButton("B3");
JButton b4 = new JButton("B4");

b3.setMaximumSize(b1.getPreferredSize());

b4.setPreferredSize(new Dimension(150, 50));
b4.setMaximumSize(new Dimension(150, 50));
```

BorderLayout

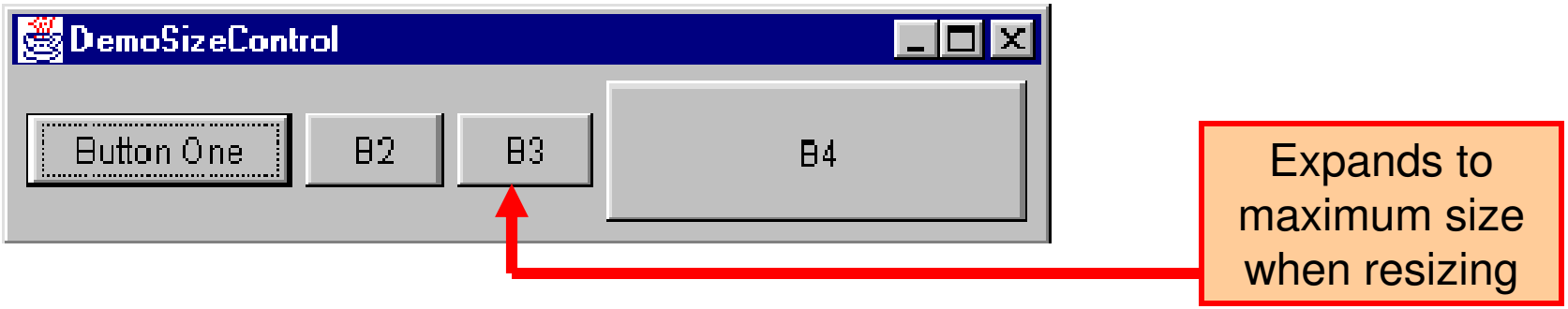


FlowLayout



GridLayout

BoxLayout

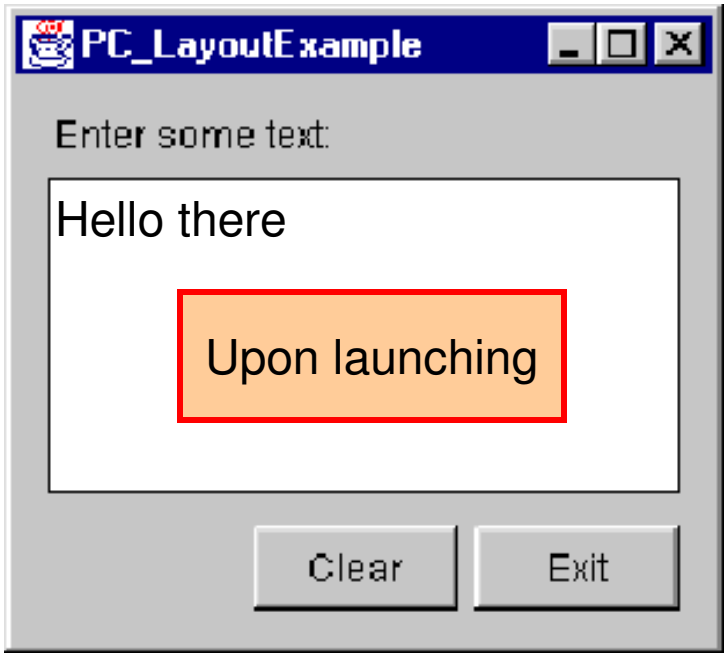


Default Layout Managers

- JPanel = FlowLayout
- JFrame’s content pane = BorderLayout

Programming Challenge

- Create a GUI layout, as follows
- Name the program
  PC_LayoutExample.java



Solution

PC_LayoutExample.java