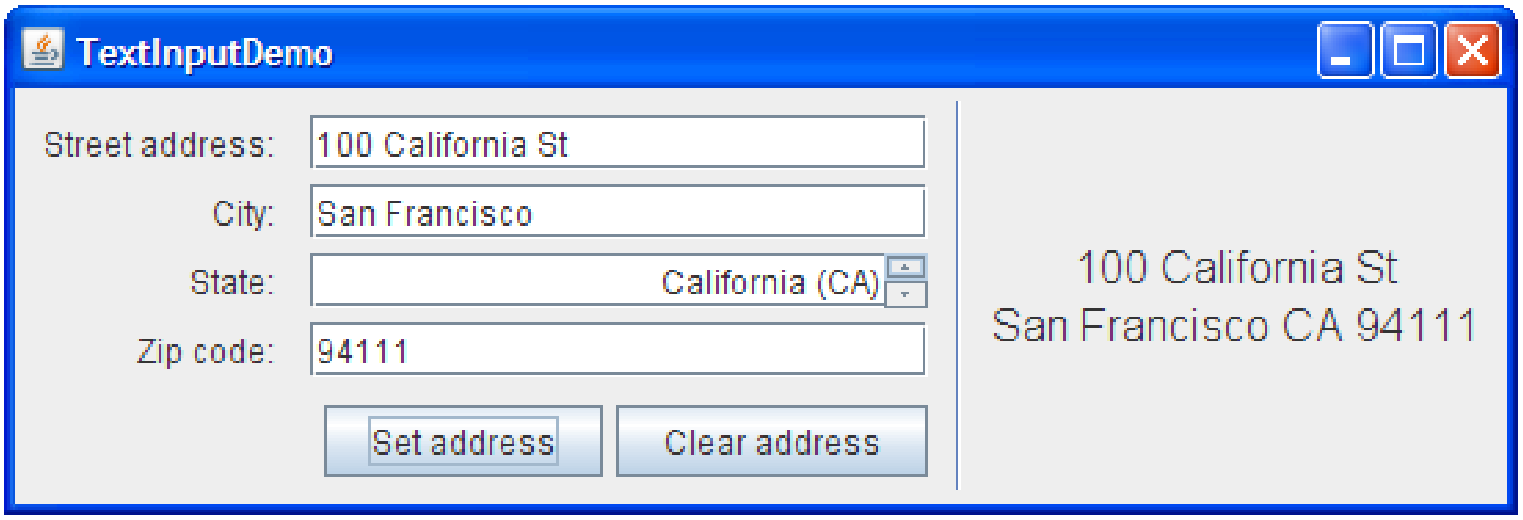
CS 106A, Lecture 25
Graphical User Interfaces (GUIs) part 3

reading:

Art & Science of Java, Chapter 10

Lecture at a glance

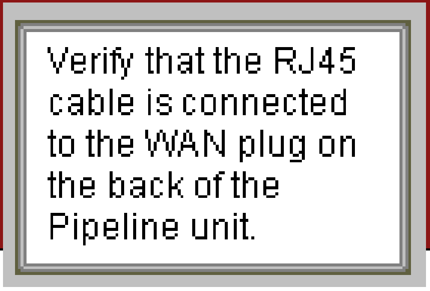
- Today we will cover more about GUIs.
  - We will see new **components** such as checxboxes and radio buttons.
- We will learn about how to do **layout**.
  - Layout allows us to position and size components in a window.
- We will learn some new types of **events**.



Street address: 100 California St
City: San Francisco
State: California (CA)
Zip code: 94111

100 California St
San Francisco CA 94111

JTextArea


Verify that the RJ45 cable is connected to the WAN plug on the back of the Pipeline unit.

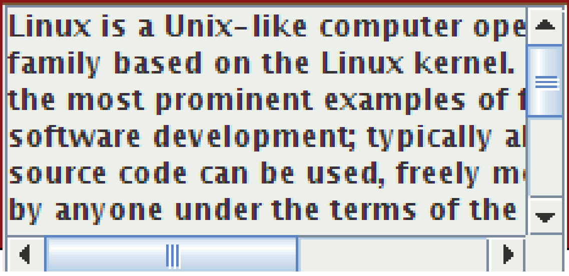
a multi-line control for typing/displaying text

Method	Description
new JTextArea("***text***") new JTextArea(***lines***, ***columns***)	Create new text area of given size
jta.getRows(), getColumns()	return dimensions of the text area
jta.isEditable() ***jta***.setEditable(***boolean***);	set/return whether user can change text
jta.setLineWrap(***boolean***);	whether long lines should wrap around
jta.setWrapStyleWord(***boolean***);	whether entire words should be kept together when doing line wrapping
jta.setCaretPosition(***index***);	sets position of typing cursor

jta.setSelectionStart(***index***);
jta.setSelectionEnd(***index***);
jta.selectAll();
	causes certain text to be selected

jta.getText() ***jta***.setText("***text***");	set/return text in the text area

JScrollPane


Linux is a Unix-like computer ope
family based on the Linux kernel.
the most prominent examples of f
software development; typically al
source code can be used, freely m
by anyone under the terms of the

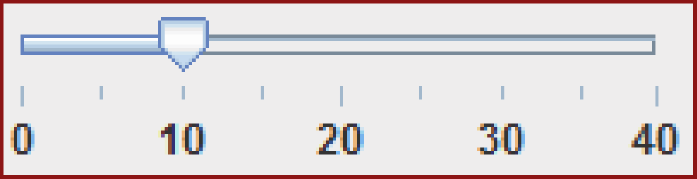
adds scrollbars around any other component

Method	**Description**
`new JScrollPane(`***`component`***`)`	creates scrollbars around the component

– After constructing the scroll pane, you must add the scroll pane, not the original component, to the window onscreen:

```
JScrollPane scroll = new JScrollPane(myTextArea);
add(scroll, CENTER);
```

JSlider



A draggable knob to choose from a range of numeric values

Method	Description
`new JSlider(`***`min`***`,` ***`max`***`,` ***`value`***`)`	construct new slider
`jsl``.addChangeListener(this);`	listen to sliding events

`jsl``.getValue()`
 `jsl``.setValue(`***`int`***`)`
	get/set current slider position

`jsl``.get/setMajorTickSpacing,`
 `jsl``.get/setMinorTickSpacing,`
 `jsl``.setPaintLabels(`***`boolean`***`),`
 `jsl``.setPaintTicks(`***`boolean`***`),`
 `jsl``.setSnapToTicks(`***`boolean`***`);`
	methods for adjusting the appearance of the slider

JSlider events

To be notified when the slider position changes:

- **1.** Modify your program class header:

```
public class MyProgram extends Program
    implements ChangeListener {
```

- **2.** Attach your program to listen to the slider.

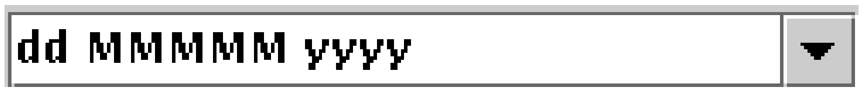
```
mySlider.addChangeListener(this);
```

- **3.** Write a **stateChanged** method to handle the event.

```
public void stateChanged(ChangeEvent event) { ... }
```

JComboBox

a drop-down list of selectable items



Method	Description
new JComboBox<***Type***>() new JComboBox<***Type***>(***array***)	construct new drop-down box that displays items of the given type
jcb.addItem("***item***");	add an item to drop-down list
jcb.getItemAt(***index***)	return item at given 0-based index
jcb.getSelectedIndex() ***jcb***.setSelectedIndex(***int***);	get/set current 0-based index of which item is selected (-1 if none selected)
jcb.getSelectedItem() ***jcb***.setSelectedItem(***item***);	get/set the text of the item that is selected (null if none selected)
jcb.isEditable() ***jcb***.setEditable(***boolean***);	get/set whether the user can type arbitrary text into the box
jcb.removeItemAt(***index***);	remove an item from the list

JList

a visible list of selectable items

Method	Description

`new JList<`***`Type`***`>()`
`new JList<`***`Type`***`>(`***`array`***`)`
	construct new list that displays items of the given type

`jl``.addItem("`***`item`***`");`	add an item to list
`jl``.getItemAt(`***`index`***`)`	return item at given 0-based index

`jl``.getSelectedIndex()`
`jl``.setSelectedIndex(`***`int`***`);`
	get/set current 0-based index of which item is selected (-1 if none selected)

`jl``.getSelectedItem()`
`jl``.setSelectedItem(`***`item`***`);`
	get/set the text of the item that is selected (null if none selected)

`jl``.setSelectionMode(`***`mode`***`);`	set whether the user can select multiple items in the list
`jl``.removeItemAt(`***`index`***`);`	remove an item from the list

JOptionPane

helper methods for displaying dialog boxes

Method	Description

`JOptionPane.showMessageDialog(`
`    this, "`***`message`***`");`
	message dialog

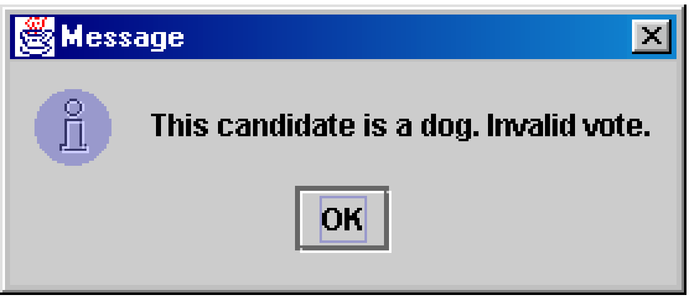
`alert("`**`message`**`");`	message dialog (Program class)

`JOptionPane.showConfirmDialog(`
`    this, "`***`message`***`")`
	yes/no or OK/cancel dialog; returns an int such as YES_OPTION

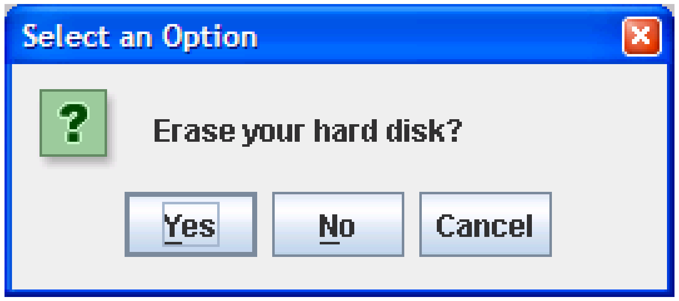
`confirm("`**`message`**`")`	yes/no dialog; returns boolean

`JOptionPane.showInputDialog(`
`    this, "`***`message`***`")`
	prompt for input with a text box; returns string typed by user

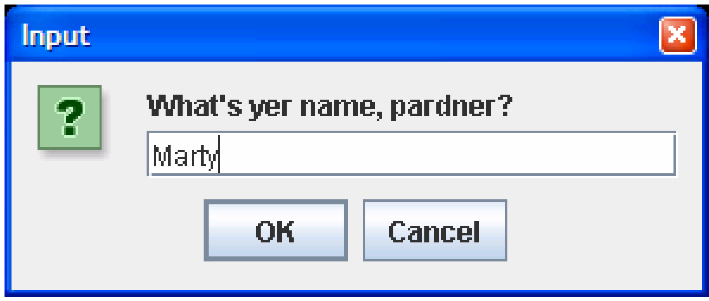
`prompt("`**`message`**`")`	prompt for input (Program class)



This candidate is a dog. Invalid vote.



Erase your hard disk?

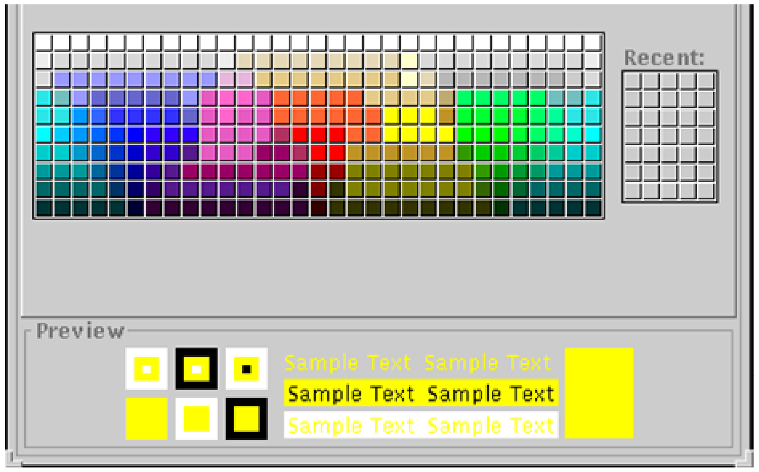


What's yer name, pardner?

Marty

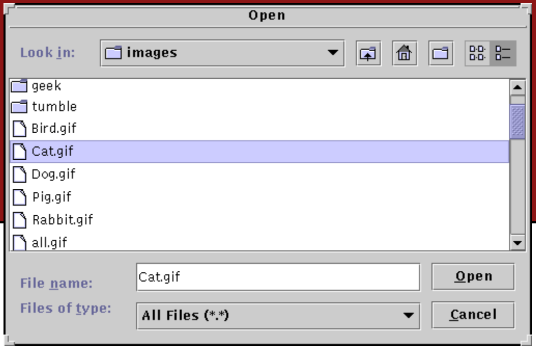
JColorChooser

a dialog box that allows the user to choose a color from a palette



- `Color color = JColorChooser.showDialog(`***`window`***`, "`***`title`***`", `***`initialColor`***`);`
  - Pops up color picker dialog, returns the color the user chose.
  - Returns `null` if user chooses the Cancel button.

JFileChooser



Look in: images

geek
tumble
Bird.gif
Cat.gif
Dog.gif
Pig.gif
Rabbit.gif
all.gif

File name: Cat.gif
Files of type: All Files (*.*)

a dialog box for browsing files

Method	Description

`new JFileChooser()`
`new JFileChooser("`***`directory`***`")`
	construct new file chooser dialog; optionally start in the given folder

`jfc``.showOpenDialog()`
`jfc``.showSaveDialog()`
	pop up the dialog to open or save a file; returns a constant indicating the result

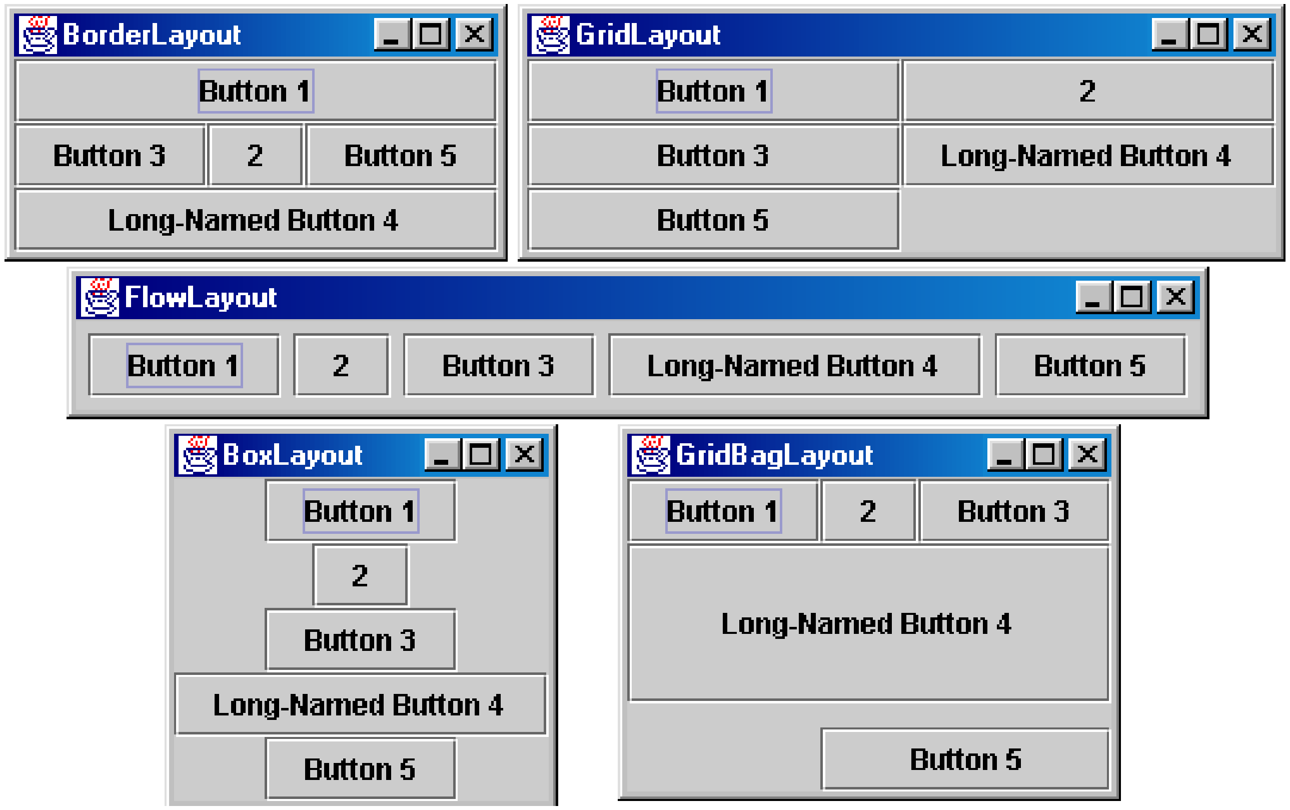
`jcb``.getSelectedFile()`	return file chosen by user (null if none)

```
JFileChooser chooser = new JFileChooser();
if (chooser.showOpenDialog() !=
        JFileChooser.CANCEL_OPTION) {
    File file = chooser.getSelectedFile();
    ...
}
```

Layout Management

Layout managers

- **Layout managers**: Objects that decide where to position each component based on some general rules or criteria.
  - "Put these four buttons into a 2x2 grid and put these text boxes in a horizontal flow in the south part of the frame."
  - Better than specifying exact pixel positions and sizes *(why?)*



Program as container

- The `Program` class acts as a **container** for holding components.

Method	Description

`add(`***`component`***`);`
`add(`***`component`***`, `***`region`***`);`
	adds a component to a container

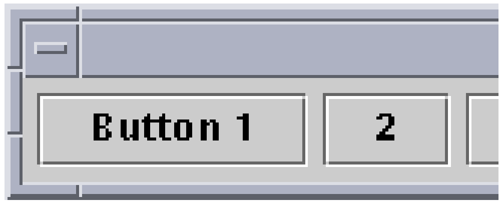
`getComponentAt(`***`index`***`)`	return component by index
`getComponentCount()`	return total number of components added
`remove(`***`component`***`);`	remove component from container
`setLayout(`***`layout`***`);`	changes container's layout strategy

Preferred size

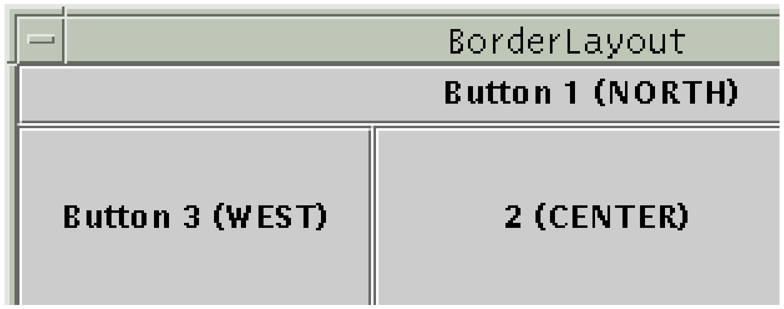
- **preferred size:** Width and height that each component would like to be, to perfectly fit its contents (text, icons, etc.).

- Some layout managers (e.g. Flow, Table) choose to respect the preferred sizes of their components as much as possible.
  - Others (e.g. Border, Grid) disregard the preferred size and use some other scheme to stretch or resize the components.

Buttons at preferred size:



Not preferred size:



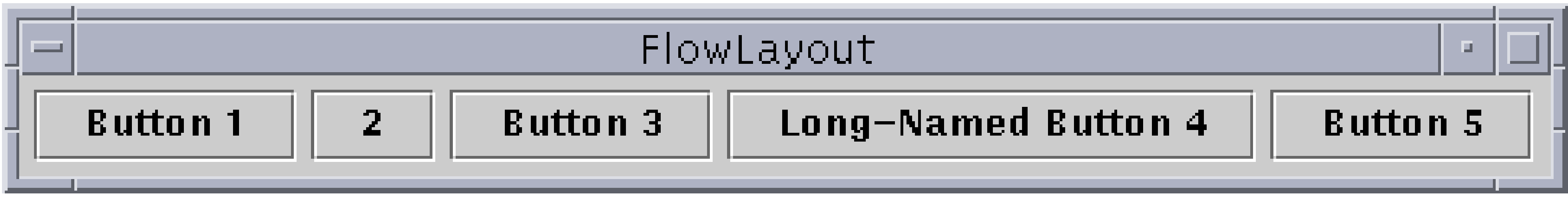
FlowLayout

treats container as a left-to-right, top-to-bottom "paragraph"

`public FlowLayout()`	constructs a new flow layout

- Components are given their preferred size, horizontally and vertically.
- Components are positioned in the order added, Left-to-Right.
- If too long, components wrap around to the next line.

```
setLayout(new FlowLayout());
add(new JButton("Button 1"));
add(new JButton("2"));
...
```



BorderLayout

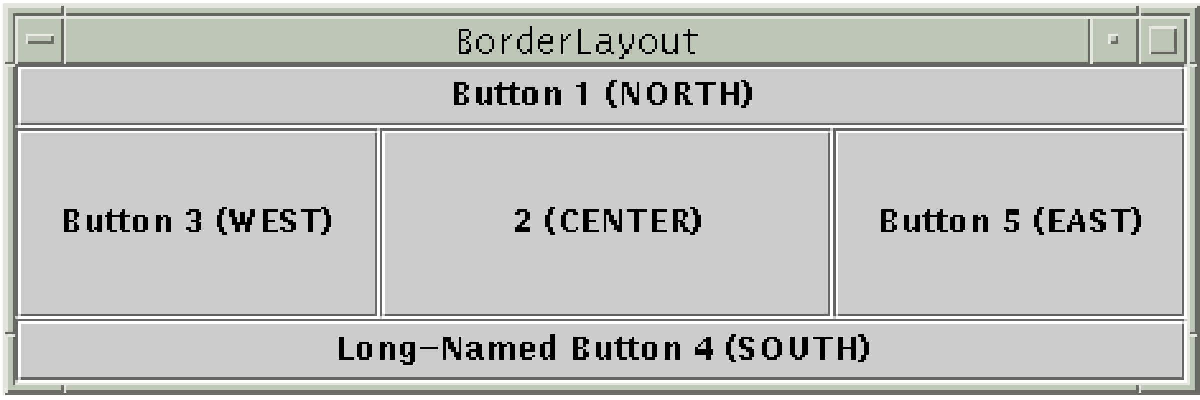
divides container into 5 regions: North, South, East, West, Center

`public` **`BorderLayout`**`()`	constructs a new flow layout

- **NORTH** and **SOUTH:** expand horizontally, preferred size vertically.
- **WEST** and **EAST** expand vertically, preferred size horizontally.
- **CENTER** expands to fill all remaining space.

```
setLayout(new BorderLayout());
add(new JButton("Button 1"), NORTH);
...
```

- the default layout for a Program

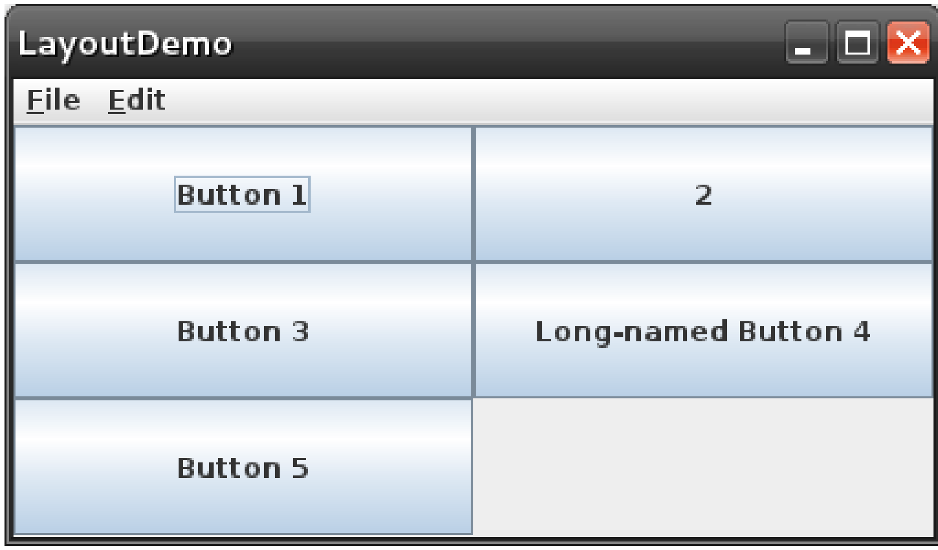


GridLayout

treats container as a grid of equally-sized rows and columns

`public GridLayout(rows, cols)`	constructs a new grid layout

- Components are given equal horizontal / vertical size.
- Completely disregards components' preferred sizes.
- Components are added in top-to-bottom, left-to-right order.
- Can specify 0 rows or columns to expand in that direction as needed.

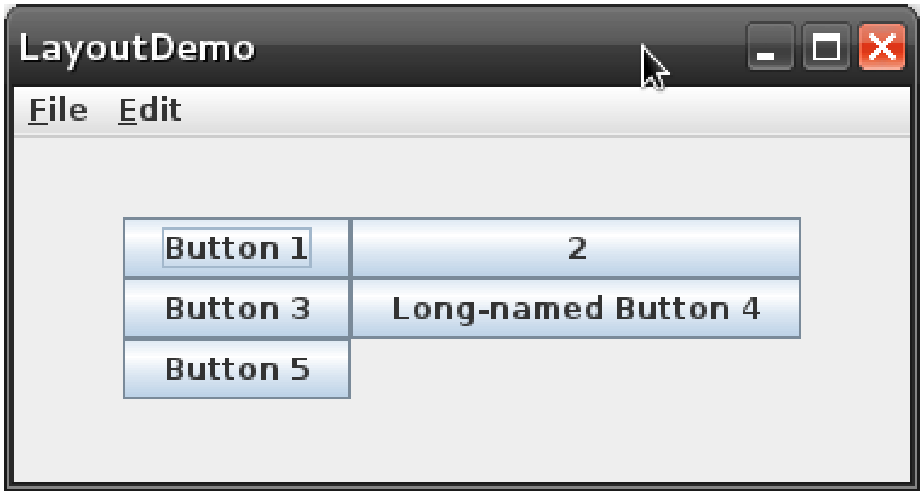


TableLayout

a grid that respects components' preferred sizes

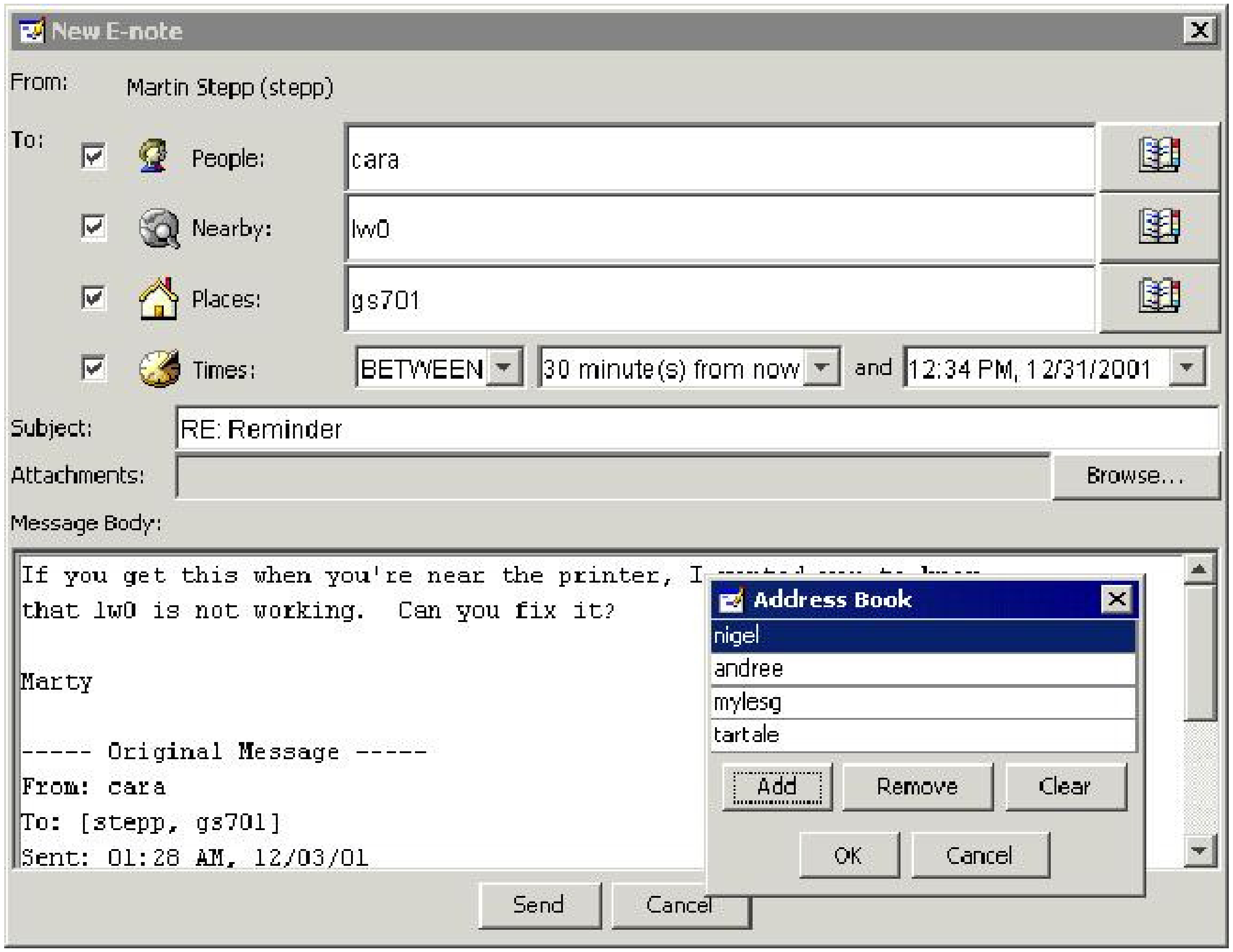
`public TableLayout(rows, cols)`	constructs a new table layout

- Treats container as a grid of rows and columns, like the grid.
- Components are sized at their *preferred size*.
- The table is centered within the overall window/container.



Complex layouts

- How would you create a large and complex GUI like this?
  - None of the layout managers shown seem powerful enough.


New E-note

From: Martin Stepp (stepp)

To: ☑ People: cara

☑ Nearby: lw0

☑ Places: gs701

☑ Times: BETWEEN 30 minute(s) from now and 12:34 PM, 12/31/2001

Subject: RE: Reminder

Attachments:

Message Body:

```
If you get this when you're near the printer, I
that lw0 is not working.  Can you fix it?

Marty

----- Original Message -----
From: cara
To: [stepp, gs701]
Sent: 01:28 AM, 12/03/01
```

Address Book: nigel, andree, mylesg, tartale

JPanel as container

- You can use a **`JPanel`** object as a **container**.
  - An invisible graphical component for containing other components.

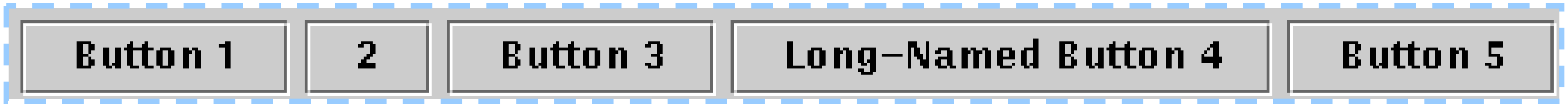
Method	Description

`new JPanel()`
`new JPanel(`***`layout`***`)`
	constructs a new panel with given layout

`jp``.add(`***`component`***`);`
`jp``.add(`***`component, region`***`);`
	adds a component to the panel

`jp``.setLayout(`***`layout`***`);`	changes panel's layout strategy

```
JPanel panel = new JPanel(new FlowLayout());
panel.add(new JButton("Button 1"));
panel.add(new JButton("2"));
...
```

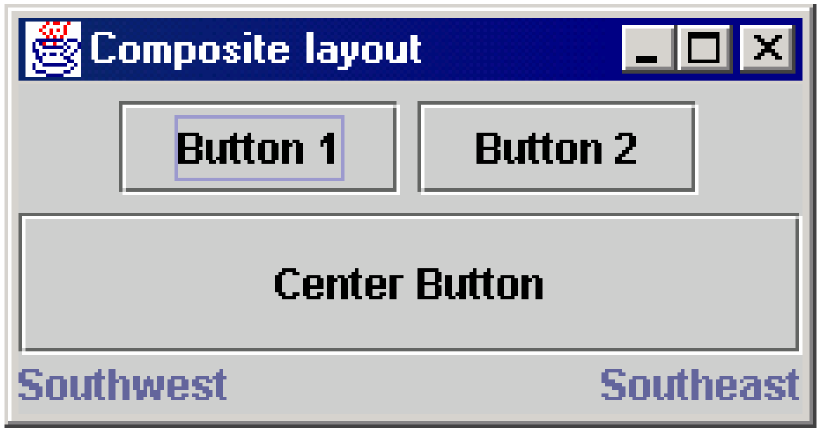


Composite layouts

- **composite layout:** One made up of containers within containers.

  – Each container has a different layout, and by combining the layouts, more complex / powerful layout can be achieved.
  – Example: A flow layout in the south region of a border layout.
  – Example: A border layout in square (1, 2) of a grid layout.

- What layouts are being used in the screenshot below?



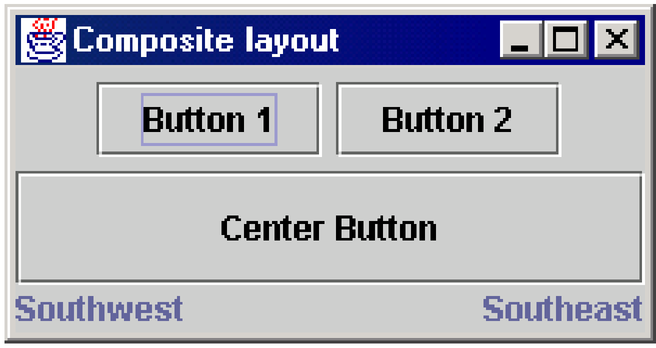
Composite layout code

```
// north area uses flow layout to position 2 buttons
JPanel north = new JPanel(new FlowLayout());
north.add(new JButton("Button 1"));
north.add(new JButton("Button 2"));

// south area uses border layout to distance 2 labels
JPanel south = new JPanel(new BorderLayout());
south.add(new JLabel("Southwest"), WEST);
south.add(new JLabel("Southeast"), EAST);

// overall panel contains the smaller panels (composite)
JPanel overall = new JPanel(new BorderLayout());
overall.add(north, NORTH);
overall.add(new JButton("Center"), CENTER);
overall.add(south, SOUTH);

// place the overall panel into the window
add(overall);
```



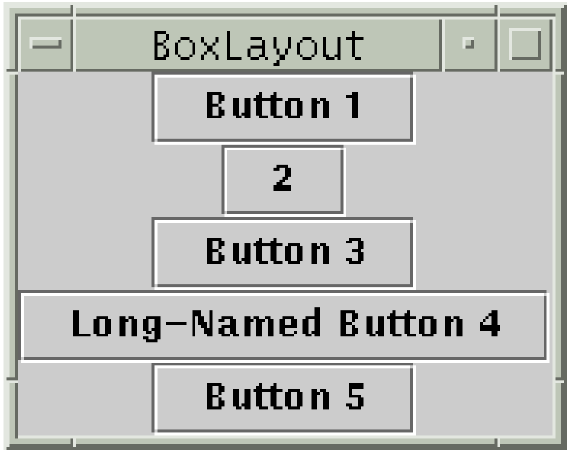
BoxLayout

a vertical flowing layout in a single column

`jp.setLayout(new BoxLayout(`
`    jp, BoxLayout.Y_AXIS));`
	constructs a new box layout

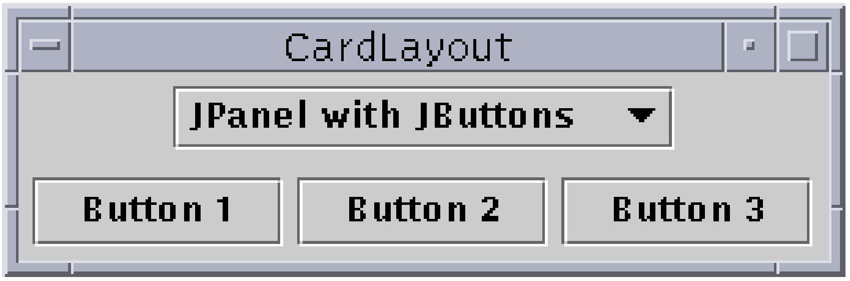
- Treats container as vertical columns; like a vertical flow layout.
- Components are sized at their *preferred size*.

```
JPanel panel = new JPanel();
panel.setLayout(new BoxLayout(
        panel, BoxLayout.Y_AXIS));
add(new JButton("Button 1"));
...
```

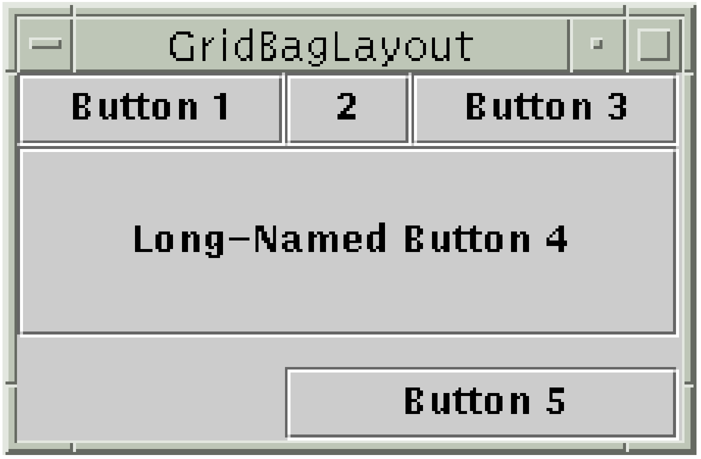


Other layouts

- **`CardLayout`**
  Layers of "cards" stacked on top of each other; one is visible at a time.



- **`GridBagLayout`**
  Powerful, but very complicated;
  Our recommendation: never use it.



- **`null`** layout (!)
  allows you to define absolute positions using setX/Y and setWidth/Height (not recommended; platform dependent)

Other Types of Events

The event hierarchy

- Java GUIs support many different event types:

Event Type	Listener	Description
`ActionEvent`	`ActionListener`	user actions on widgets, e.g. buttons
`ComponentEvent`	`ComponentListener`	changes to a component, e.g. position, size
`ContainerEvent`	`ContainerListener`	changes to a container's contents
`FocusEvent`	`FocusListener`	component gains/loses keyboard focus
`InputEvent`	`InputListener`	various types of user input
`KeyEvent`	`KeyListener`	user presses keys on a component
`MouseEvent`	`MouseListener`	user moves/clicks mouse on a component
`PaintEvent`	`PaintListener`	component's pixels are painted on screen
`TextEvent`	`TextListener`	text of a component changes, e.g. text field
`WindowEvent`	`WindowListener`	window size/location/status changes

Event listening

- **1.** Modify your program class header:

```
public class MyProgram extends Program
    implements TypeListener {
```

- **2.** Add your program to listen to events.

```
myComponent.addTypeListener(this);
```

- **3.** Write any necessary methods to handle the events.

```
public void methodName(TypeEvent event) { ... }
```

  – Consult the Java API documentation to learn necessary methods.

Example: WindowListener

```
public class MyProgram extends Program
        implements WindowListener {

    public void init() {
        addWindowListener(this);
    }

    public void windowClosing(WindowEvent e) {
        alert("Window is closing now!");
    }
}
```

Example: KeyListener

```
public class MyProgram extends Program
        implements KeyListener {
    private JTextField field;

    public void init() {
        ...
        field.addKeyListener(this);
    }

    public void keyTyped(KeyEvent e) {
        alert("You pressed a key!");
    }
}
```

Overflow (extra) slides

JMenuBar

a drop-down menu of commands

- `public` **`JMenuBar`**`()`
- `public void` **`add`**`(JMenu menu)`

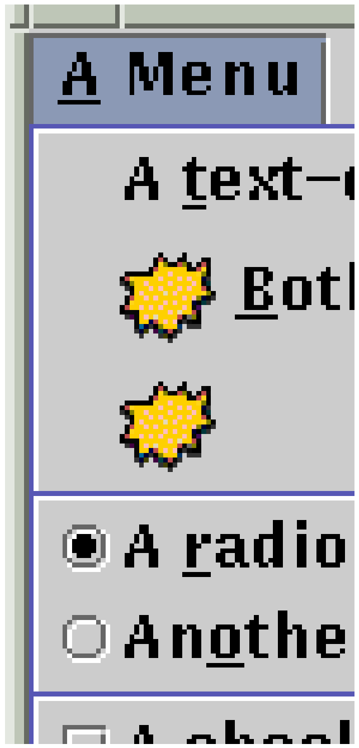
Usage: in Program class, the following method exists:

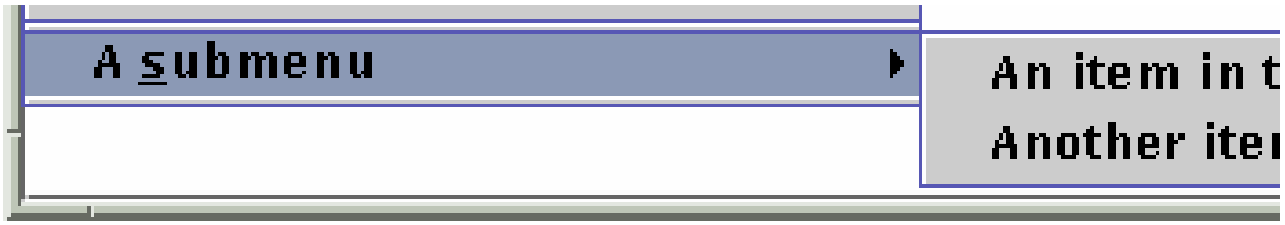
- `public void` **`setJMenuBar`**`(JMenuBar bar)`

JMenu

a sub-menu of commands with a JMenuBar

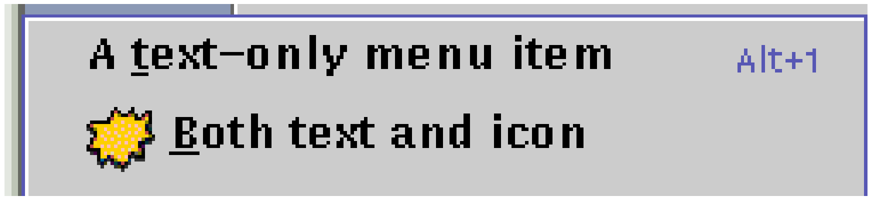
- `public` **`JMenu`**`(String text)`
- `public void` **`add`**`(JMenuItem item)`
- `public void` **`addSeparator`**`()`
- `public void` **`setMnemonic`**`(int key)`





JMenuItem

an entry within a JMenu that can be clicked to execute a command



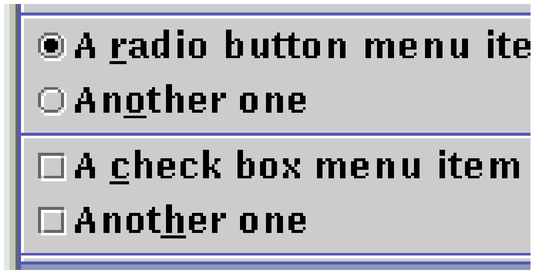
- public **JMenuItem**(String text)
- public **JMenuItem**(String text, Icon icon)
- public **JMenuItem**(String text, int mnemonic)

- public void **setEnabled**(boolean b)

- public void **addActionListener**(ActionListener al)

J(CheckBox|RadioButton)MenuItem

a JMenuItem with a check box or radio circle



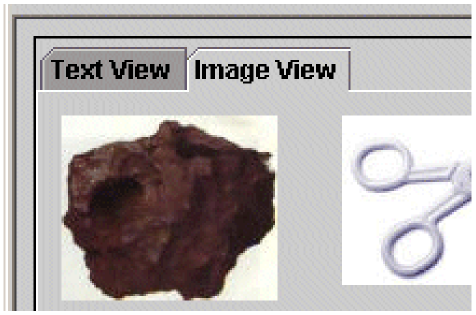
- `public J_____MenuItem(String text)`
- `public J_____MenuItem(String text, boolean selected)`
- `public J_____MenuItem(String text, Icon icon)`
- `public J_____MenuItem(String text,`
  `Icon icon, boolean selected)`
- `public void addActionListener(ActionListener al)`
- `public boolean isSelected()`
- `public void setSelected(boolean b)`

Recall: in a **ButtonGroup**, the following method exists:

- `public void add(button)`

JTabbedPane

a container that holds subcontainers, each with a "tab" label and content

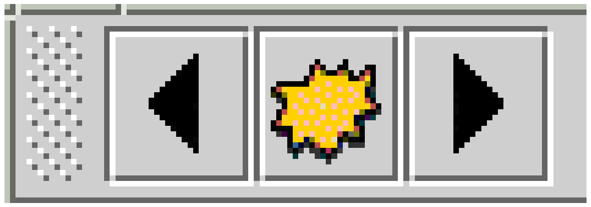


- `public` **`JTabbedPane`**`()`
  `public` **`JTabbedPane`**`(int tabAlignment)`
  Constructs a new tabbed pane. Defaults to having the tabs on top; can be set to `JTabbedPane.BOTTOM`, `LEFT`, `RIGHT`, etc.

- `public void` **`addTab`**`(String title, Component comp)`
- `public void` **`insertTab`**`(...)`
- `public void` **`remove`**`(Component comp)`
- `public void` **`remove`**`(int index)`
- `public void` **`removeAll`**`()`
- `public void` **`setSelectedComponent`**`(Component c)`
- `public void` **`setSelectedIndex`**`(int index)`

JToolbar

a movable dock container to hold common app buttons and commands



- `public` **`JToolBar`**`()`
- `public` **`JToolBar`**`(int orientation)`
- `public` **`JToolBar`**`(String title)`
- `public` **`JToolBar`**`(String title, int orientation)`
  Constructs a new tool bar, with optional title and orientation; can be `JToolBar.HORIZONTAL` or `VERTICAL`, default horizontal

- `public void` **`add`**`(Component comp)`
  Adds the given component to this tool bar.
  - Note: If using JToolbar, don't put other components in N/E/S/W.