Swing Widgets and Layout

10 February 2011
CMPT166
Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

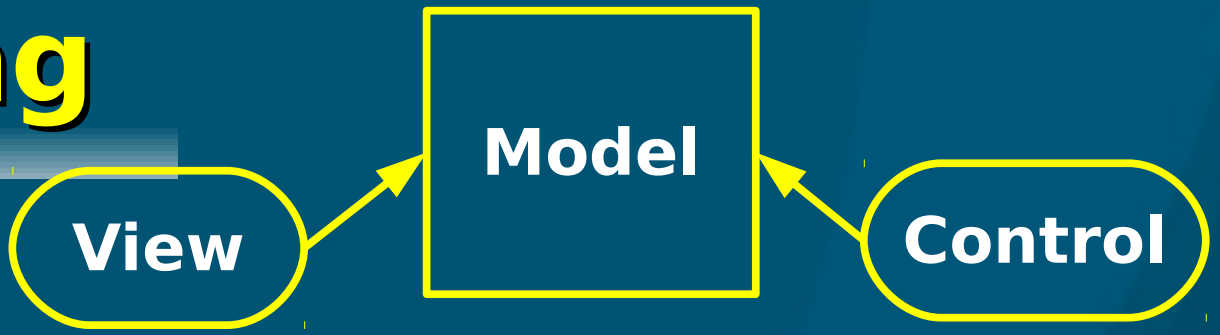
What's on for today

- MVC design of Swing
- Class hierarchies of events and listeners
- Swing widgets
  - JLabel, JTextField, JButton, JCheckBox, JRadioButton, JComboBox
- Layout managers
  - Flow, Border, Grid, GridBag, Group

Model-View-Controller

- Design patterns: reusable, generic concepts to help you design your programs
- MVC design pattern:
  - Model: stores data
    - Computation, methods to transform data
    - Data structure issues: arrays? Linked-lists? Classes?
  - View: display / output / read
    - println()? Swing? Web? JTextField?
  - Controller: manipulate / input / write
    - Command-line? Buttons? Mouse?

MVC in Swing

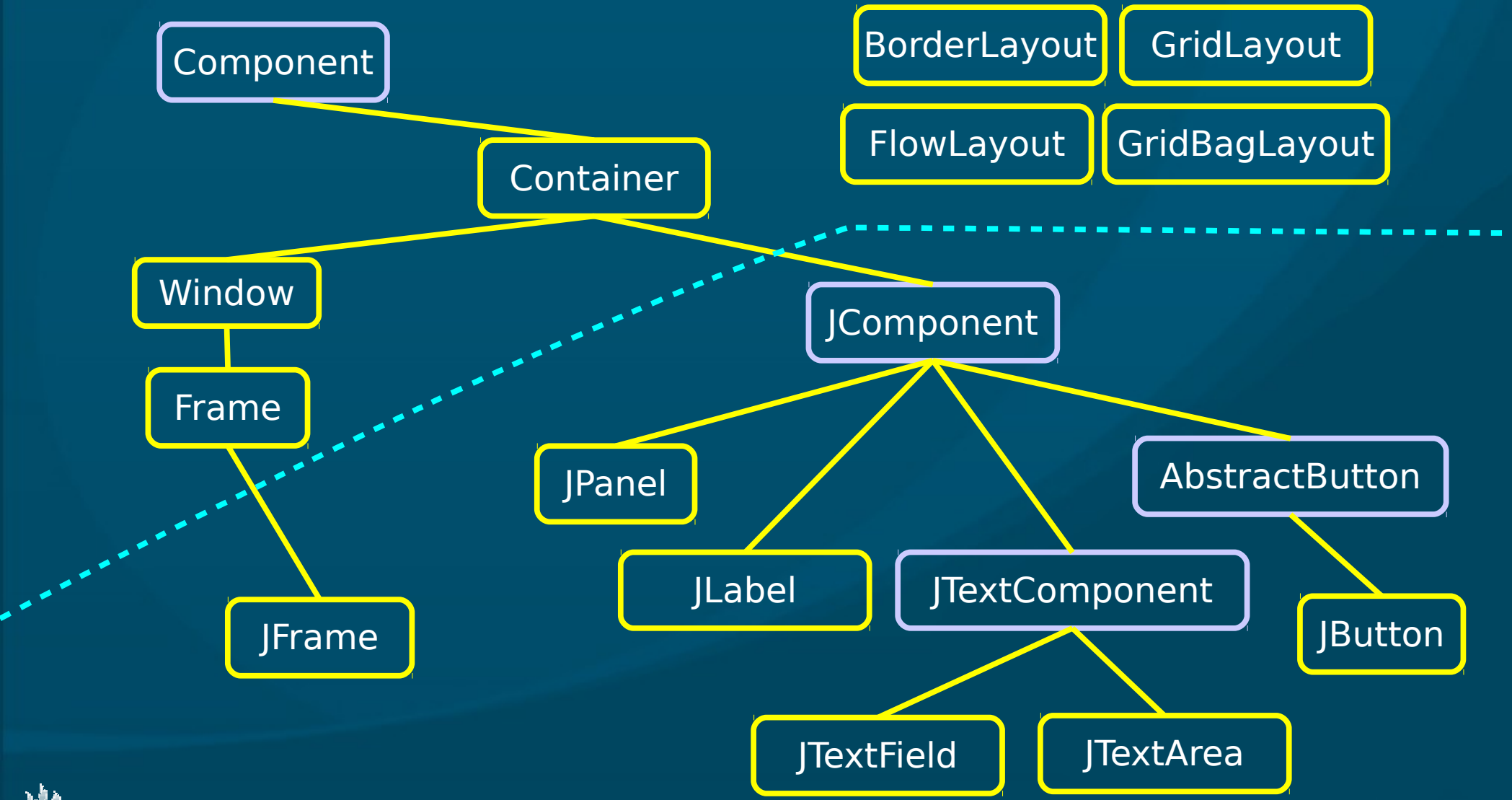


- Model:
  - Core content/functionality of program
  - Ideally, should be independent of Swing
- View:
  - JFrame, JPanel, layout manager, widgets
- Controller: Event handler:
  - implements ActionListener, ItemListener {
    - public void actionPerformed(ActionEvent e)
    - public void itemStateChanged(ItemEvent e)

Swing container classes

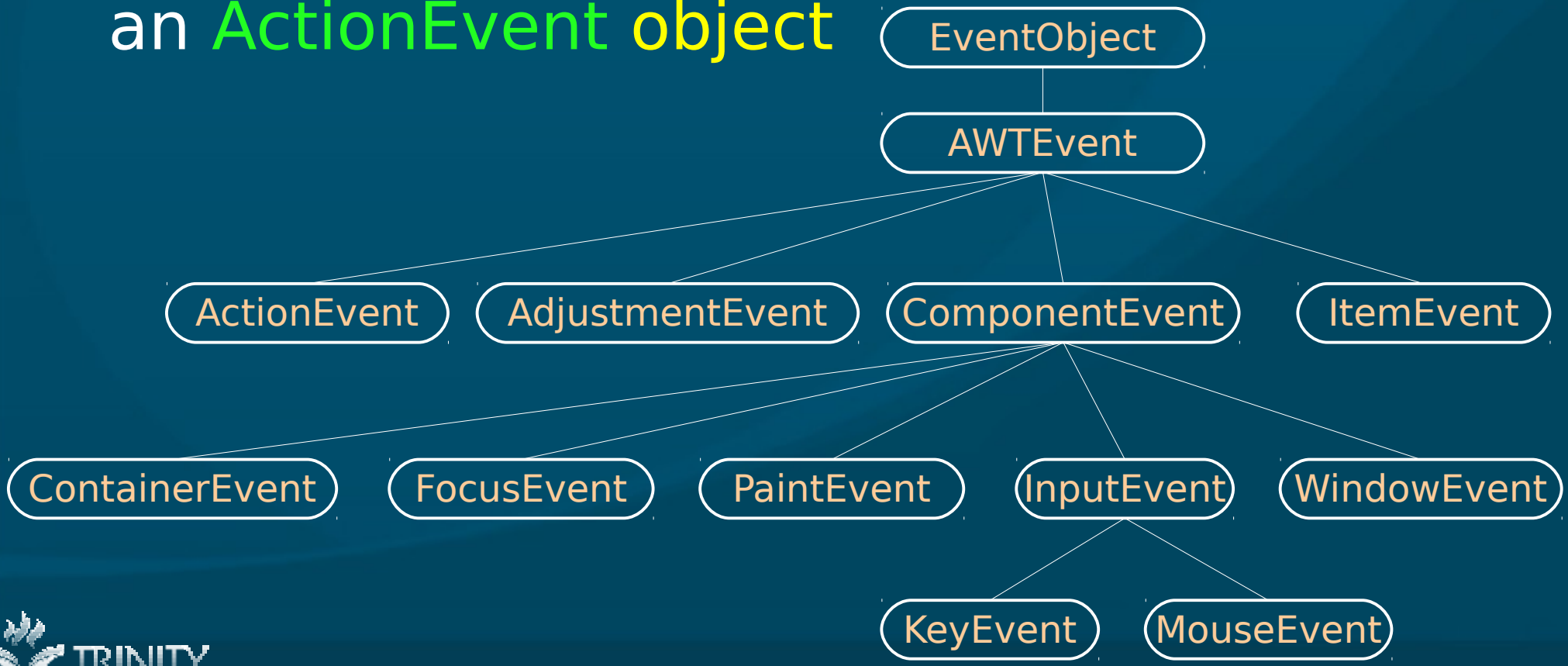
- Containers (java.awt.Container) hold other components
  - Swing containers: javax.swing.JComponent
    - e.g., both JFrame and JPanel
  - Every JComponent can have one layout manager: decides how to arrange widgets
- JFrame: Swing window
  - Can only have one layout manager
  - But can nest JPanels, and each JPanel can have its own layout manager

Swing / AWT class hierarchy



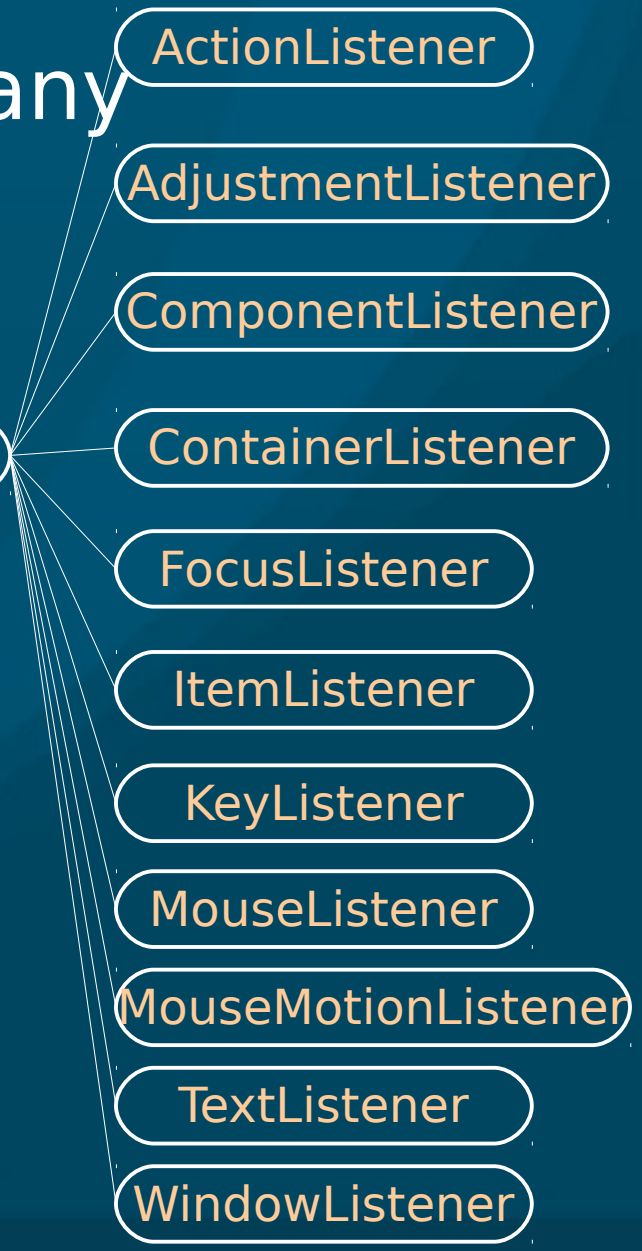
Types of events

- Event classes are in package java.awt.event
- e.g., the ActionListener interface uses the actionPerformed() method on an ActionEvent object



EventListener interfaces

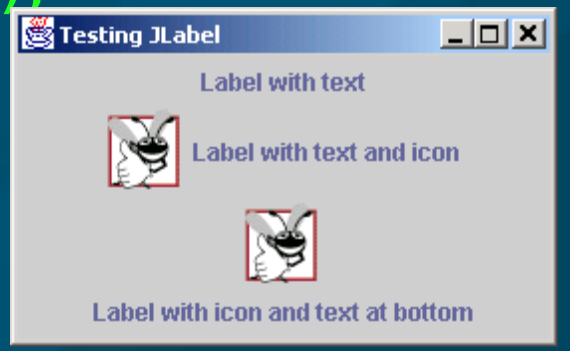
- ActionListener is but one of many interfaces for handling events
- KeyListener: KeyEvent
  - Listen for keypresses
- MouseListener: MouseEvent
  - Press/release, enter/exit
- MouseMotion: MouseEvent
  - Move, drag



JLabel

- Intended to be a text/image widget describing another component

```
Label1 = new JLabel( “Rotation” );
```



  - Change the text:

```
label1.setText( “Rot” );
```

  - Add a tooltip:

```
label1.setToolTipText( “Rotation in degrees” );
```

  - Add an icon:

```
Icon rotIcon = new ImageIcon( “rot.gif” );
label1.setIcon( rotIcon );
```

Text fields

- JTextField:
  - Single-line widget for user to type in text

```
text1 = new JTextField( 10 );   // field width
text2 = new JTextField( “Type your name here” );
```

  - Read or change the text in the box with .getText() and .setText(String s)
  - Disable user editing:

```
text1.setEditable( false );
```

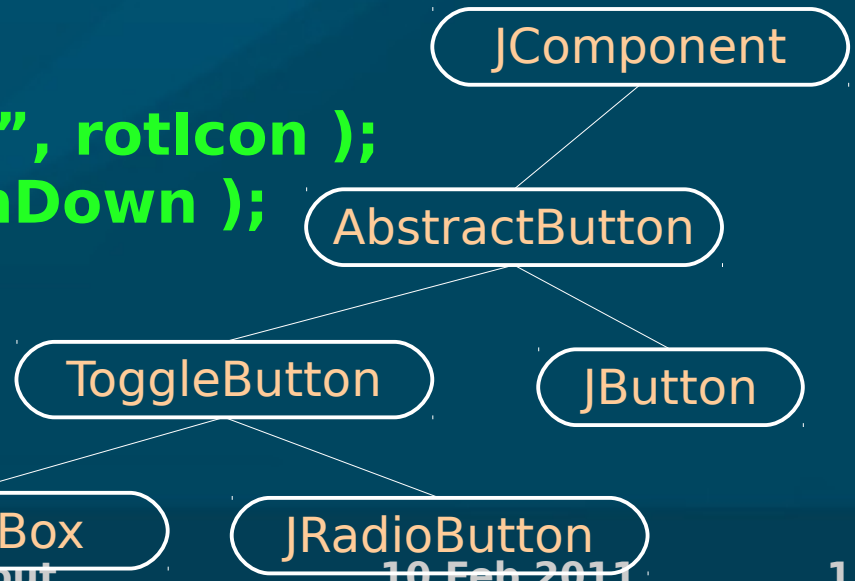
- JPasswordField: subclass, shows only dots
- JTextArea: allows multiple lines, word-wrap

JButton

- User clicks to trigger an ActionEvent
- Several types:
  - Command button, check box, toggle, radio
- Abstract superclass: AbstractButton

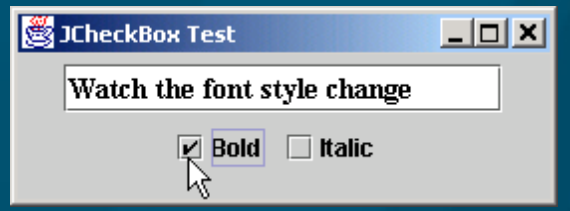
```
Icon rotIcon = new ImageIcon( “rot.png” );
Icon rotIconDown = new ImageIcon( “rotdn.png” );

rotButton = new JButton( “Rotate”, rotIcon );
rotButton.setRolloverIcon( rotIconDown );
```



JCheckBox and ItemListener

- JCheckBox uses a diff listener interface:

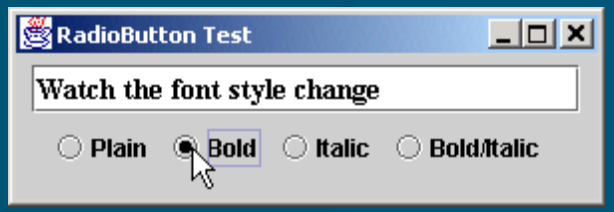



```
boldCheck = new JCheckBox( “Bold” );
boldCheck.addItemListener( ... );
```

- ItemListener interface uses itemStateChanged() method on an ItemEvent object:

```
boldCheck.addItemListener( new ItemListener() {
    public void itemStateChanged( ItemEvent event ) {
        if ( event.getStateChange() == ItemEvent.SELECTED ) {
                ...
        }
    } } );
```

JRadioButton




```
plainRadio = new JRadioButton( “Plain”, false );
boldRadio = new JRadioButton( “Bold”, true );
italicRadio = new JRadioButton( “Italic”, false );
```

- JRadioButton also uses ItemListener:

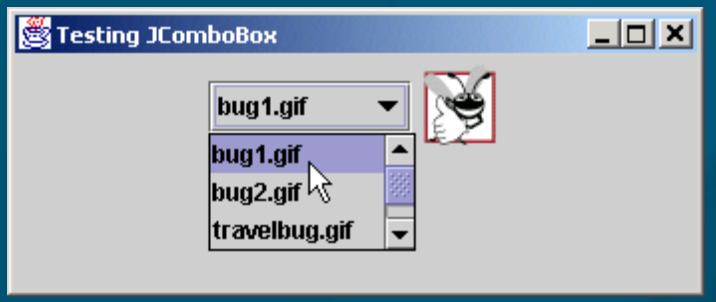
```
boldRadio.addItemListener( new ItemListener() {…} );
```

- Usually put radio buttons in a ButtonGroup:

```
geomGroup = new ButtonGroup();
geomGroup.add( plainRadio );
geomGroup.add( boldRadio );
geomGroup.add( italicRadio );
```

- Also add() buttons to the window/panel

JComboBox



- Drop-down list;
  user can choose only one entry

```
private String geom[] =
        { “Triangles”, “Quads”, “Tristrips” };
geomCombo = new JComboBox( geom );
```

- Show only three rows at a time:

```
geomCombo.setMaximumRowCount( 3 );
```

- Also uses ItemListener interface
- See which entry user selected (0, 1, 2, etc.):

```
geomCombo.getSelectedIndex()
```

Layout managers

Ref: Java tutorial

- Position widgets within the JPanel/JFrame
  - The panel calls setLayout():
    - setLayout(new FlowLayout());
  - Then widgets are then add()ed to the panel in order:
    - add(widget1);
- FlowLayout: simple left-to-right
- BorderLayout: along the edges
- GridLayout: regular grid of equal-size cells
- GridBagLayout: table of unequal-size cells
- GroupLayout: hierarchical grouping in each axis

FlowLayout

- Default and simplest
- Simple left-to-right horizontal arrangement
- Widgets laid out in the order they were add()ed:

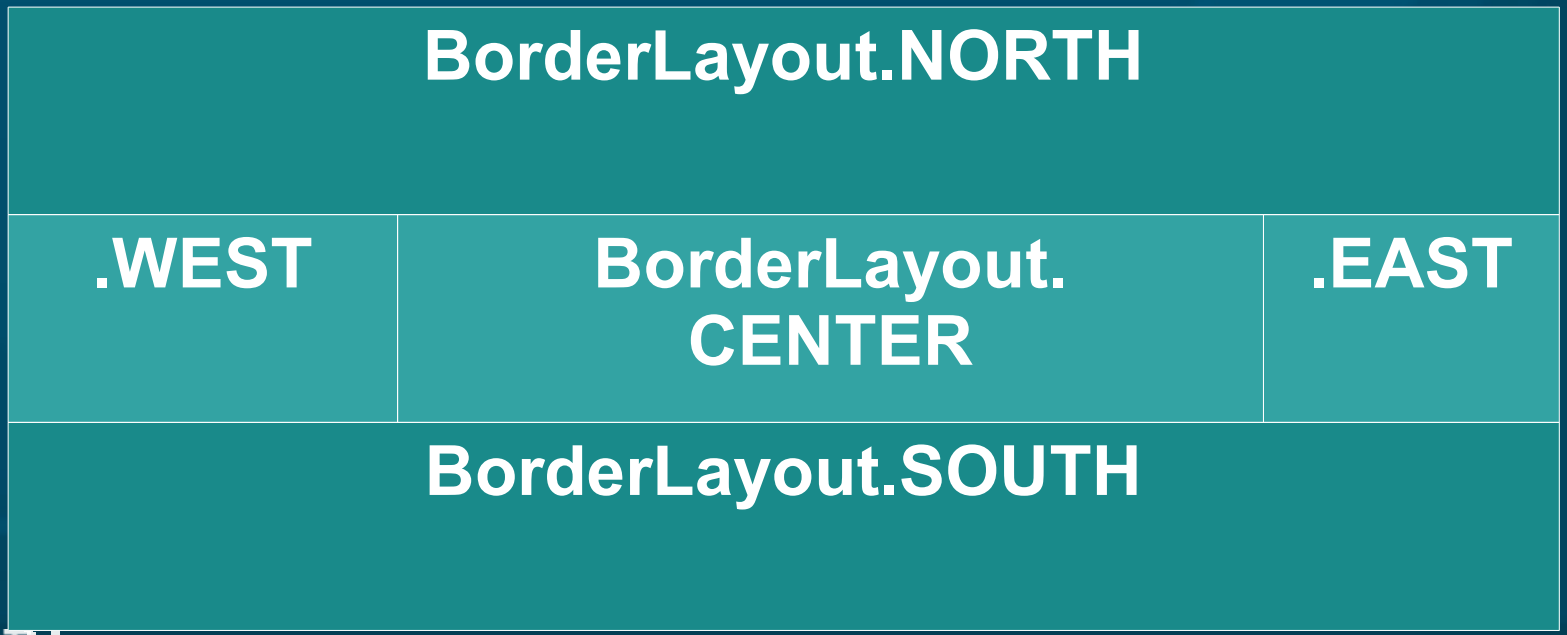
```
add( widget1 );
add( widget2 );
```

- If not enough horizontal space for widgets, flow continues on next row
- Can setComponentOrientation() to layout from right-to-left instead

BorderLayout

- Position widgets along edges of the panel
- Often used to organize sub-panels
- Edges: north, south, east, west, center

```
add( widget1, BorderLayout.NORTH );
```



GridLayout

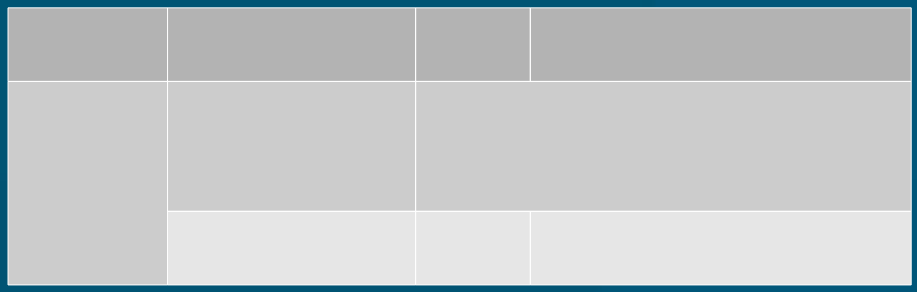
- Uses a 2D grid (table) of equal-size cells
- Constructor specifies number of rows, cols:

```
setLayout( new GridLayout(2, 3) );
```

- Widgets are added in order, from top-left cell across to top-right, then filling each row
  - If too many widgets, adds extra columns

1	2	3
4	5	6

GridBagLayout



- Cells of a rectangular grid, but not all equal size
- Components can also span multiple cells
- More flexible, but more complex: c.f. HTML tables
- Specify location of each widget via constraints:

```
GridBagConstraints c = new GridBagConstraints();
c.gridx = 0; c.gridy = 1; c.gridheight = 2;
add( widget1, c );
```

  - Optional weights indicate relative space to occupy (e.g., for resizing)

```
c.weightx = 0.2;          // get less space
```

GroupLayout

- Used in visual GUI designer: NetBeans Matisse
- Specify horizontal and vertical axes separately
- Specify groups:
  - Sequential (left-to-right / top-to-bot) or
  - Parallel (aligned on top of each other)
- In pseudocode:
- **x: Seq(w1, w2, Par(w3, w4))**
- **y: Seq(Par(w1, w2, w3), w4)**