Swing: Layout Managers

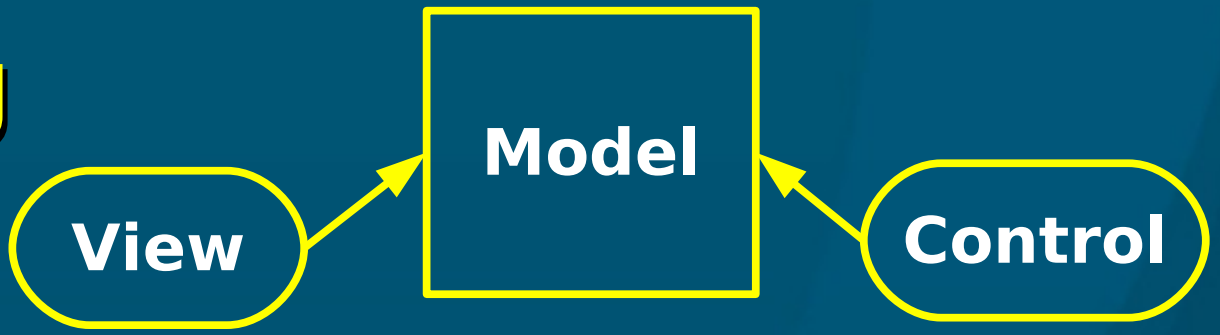
12 Feb 2010
CMPT166
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

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

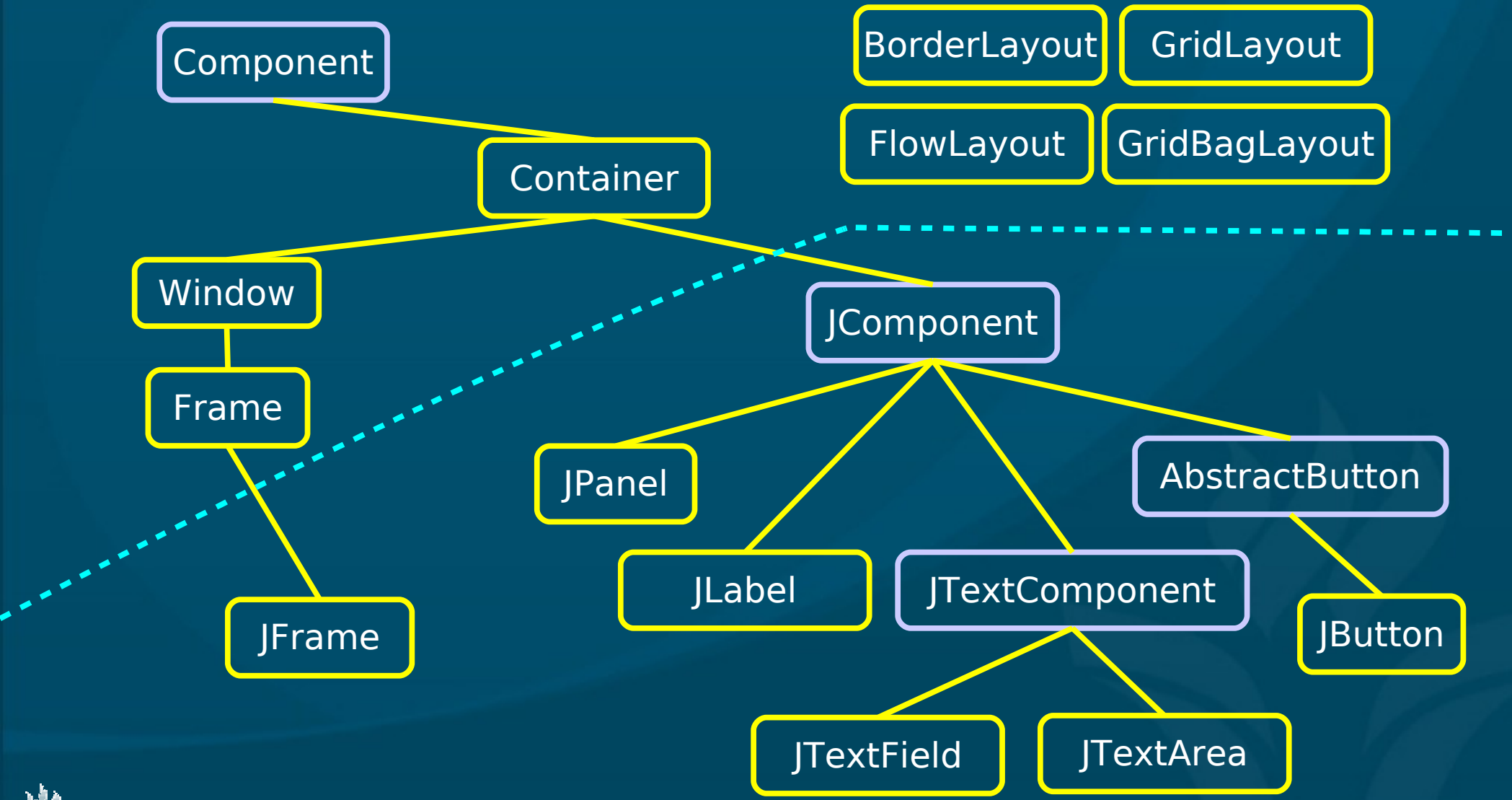


- Model:
  - Core content/functionality of program
  - Ideally, should be independent of Swing
- View:
  - JFrame, JPanel, layout manager, widgets
- Controller: Event handler:
  - implements ActionListener, ItemListener {
    - public void actionPerformed(ActionEvent e)
    - public void itemStateChanged(ItemEvent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



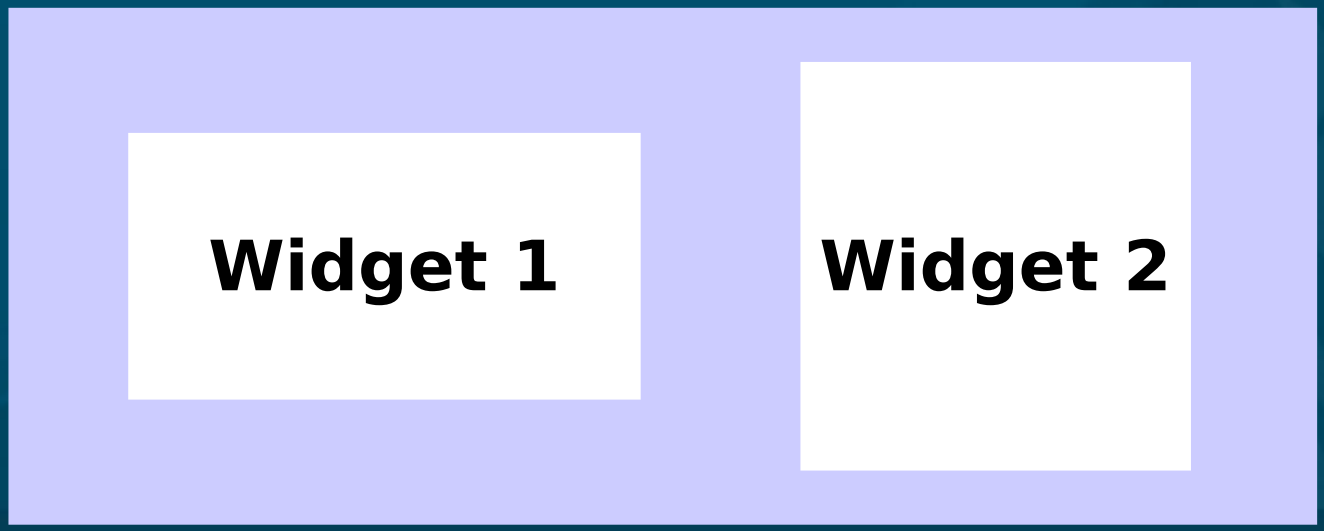
Layout managers

Ref: Java tutorial

- Positions widgets within the JPanel/JFrame
  - The panel calls setLayout():
    - **setLayout(new FlowLayout());**
  - Then widgets are add()ed to the panel:
    - **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

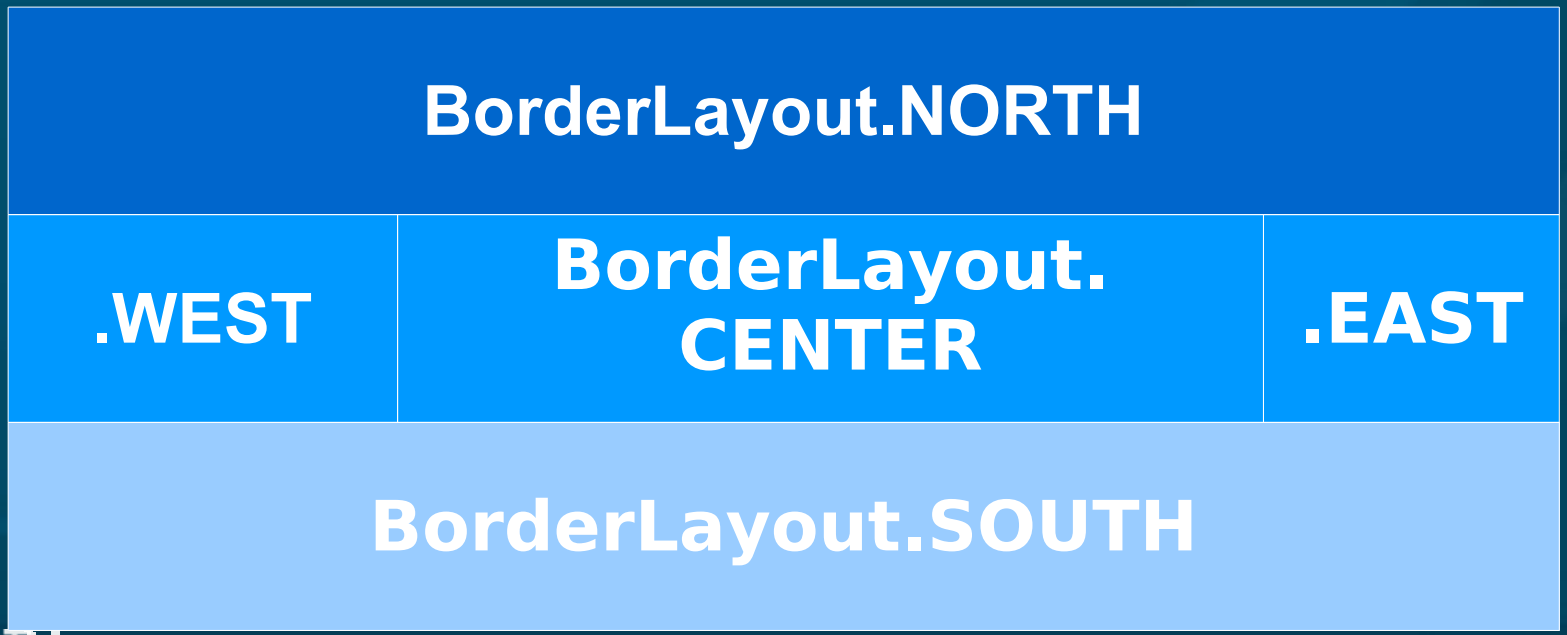
FlowLayout

- Default and simplest layout
- Simple left-to-right horizontal arrangement
- Widgets laid out in the order they were add()ed
- If not enough space, flow continues on next row
- Can setComponentOrientation() to right-to-left



BorderLayout

- Position widgets along the edges of the panel
- Often used to organize sub-panels
- Edges: north, south, east, west, center
  - `add( widget1, BorderLayout.NORTH );`

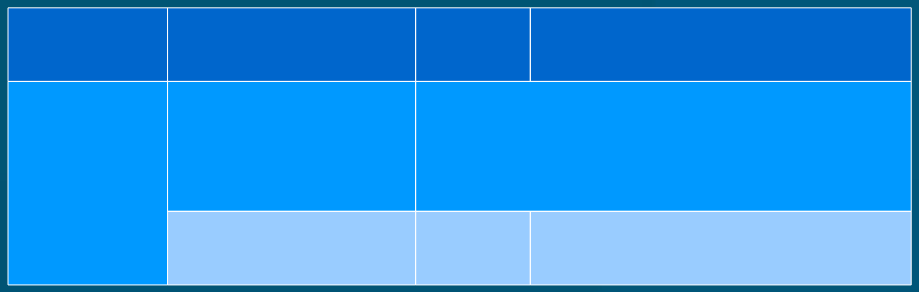


GridLayout

- 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

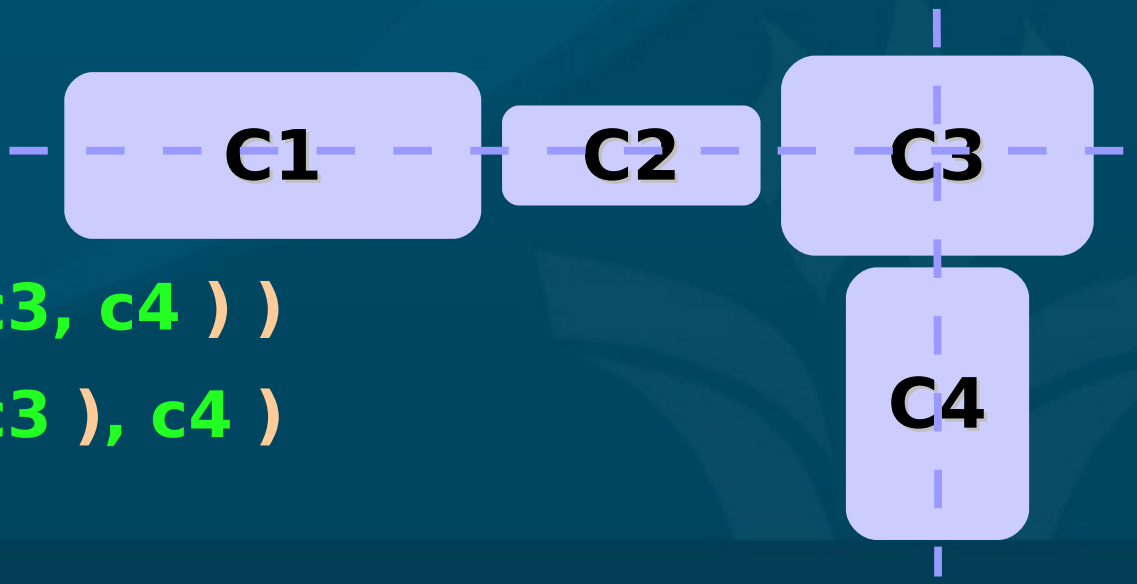
GridBagLayout



- Cells of a rectangular grid, but not all equal size
- Components can also span multiple cells
- More flexible, but more complex: cf HTML tables
- Specify location of each widget via constraints:
  - **GridBagConstraints c = new GridBagC...();**
  - **c.gridx = 0; c.gridy = 1; c.gridheight = 2;**
  - **add(widget1, c);**
  - May include weights indicating relative amount of space to occupy: e.g., for resize
    - **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(c1, c2, Par(c3, c4))**
- **y: Seq(Par(c1, c2, c3), c4)**