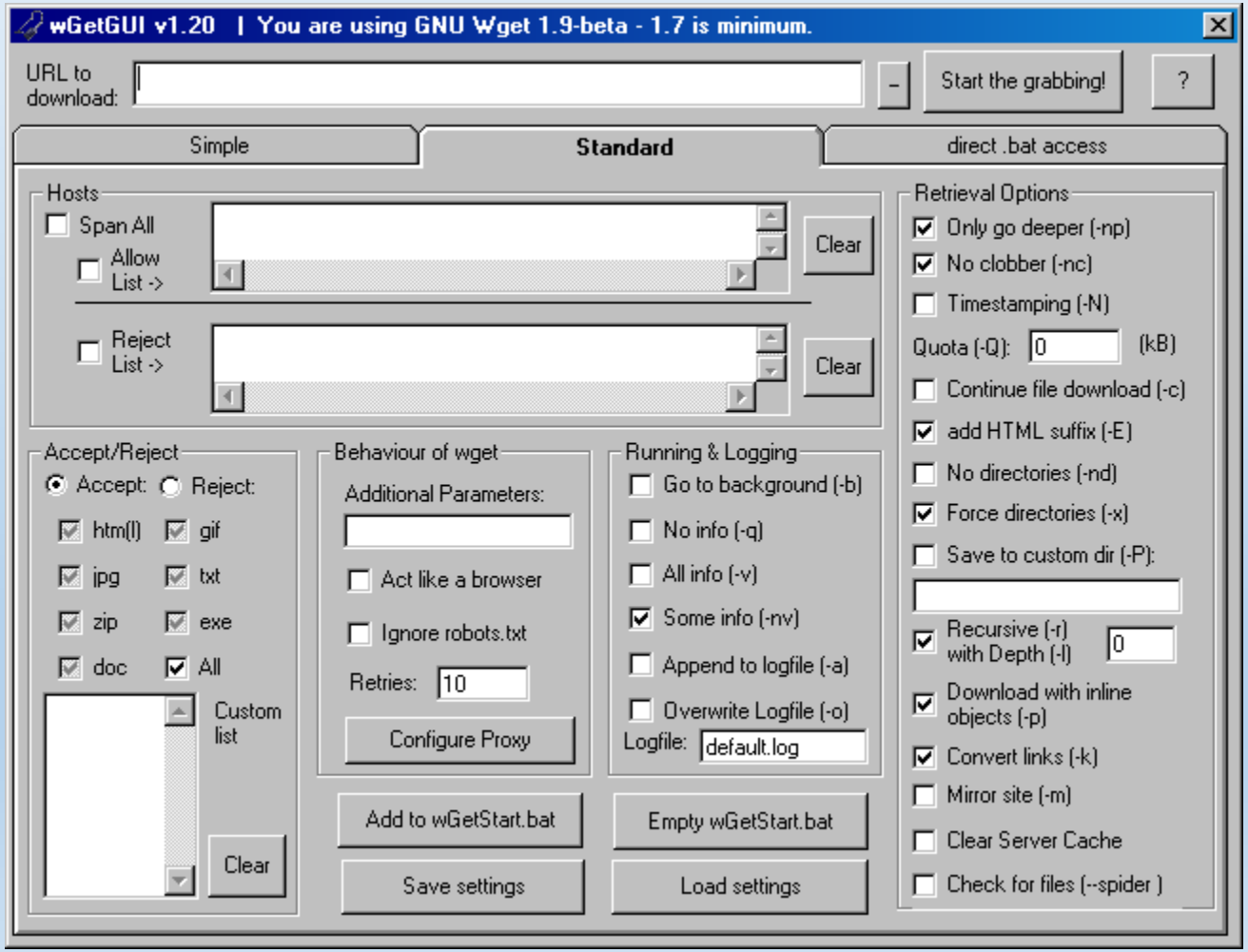
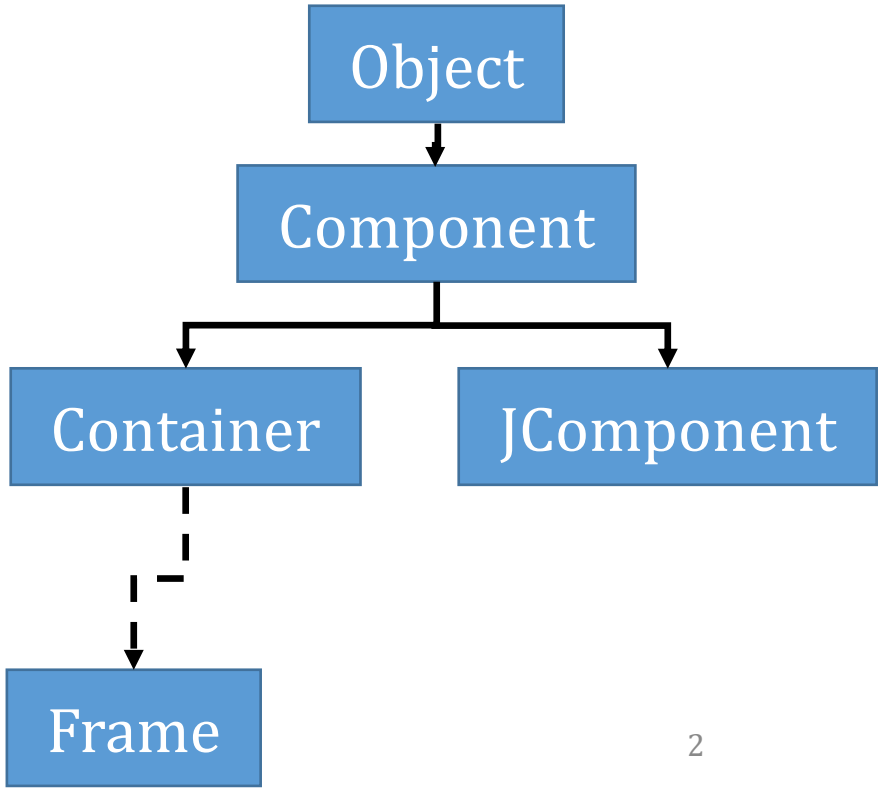
Swing Containers



Chapter 20.1

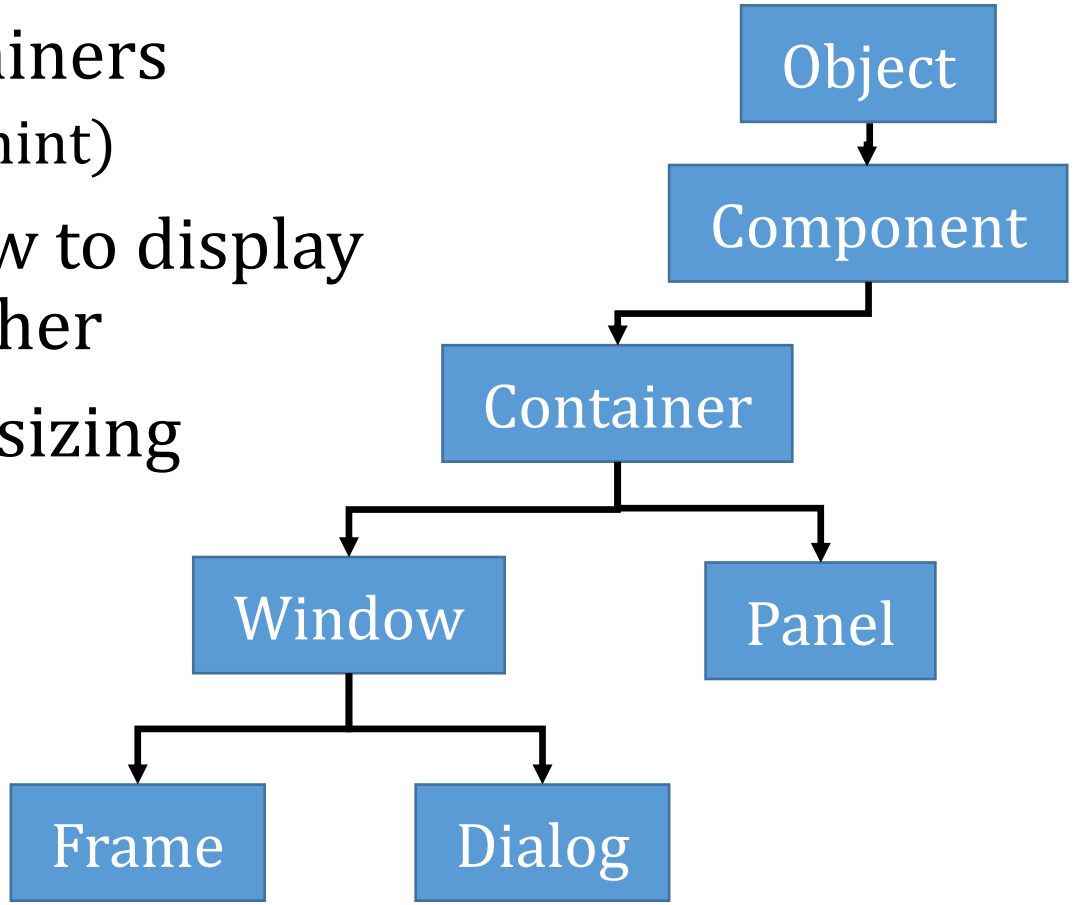
Swing is Rectangular

- You get a rectangular window from the X server or "Frame"
- You fill up that rectangular window with rectangular components
  - Think of these as two dimensional "blocks"
- The basic class in Swing is "Component"
  - public void setSize(int width,int height)
- There are two kinds of components:
  - Container (Contains other components)
  - JComponent (These are widgets)



Containers

- We put components inside containers
  - container.add(component,layout-hint)
- We need to tell the container how to display components relative to one another
- We need to manage container resizing
  - Layout Management



Managing the Layout Manager

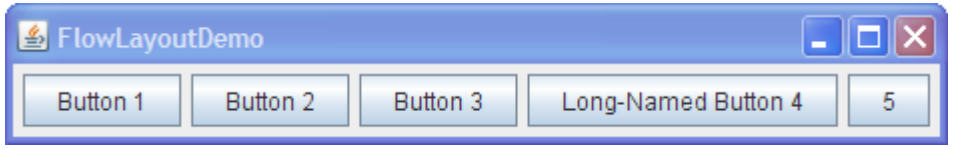
- Layout manager may be an argument to a Container Constructor
  - public JPanel(LayoutManager layout)
  - e.g. JPanel panel = new JPanel(new BorderLayout());

  Create a new JPanel with a "border" layout manager instead of "flow"

- Contain class has a setLayout (and a getLayout) method
  - public void setLayout(LayoutManager mgr)

  Sets the layout manager for this container

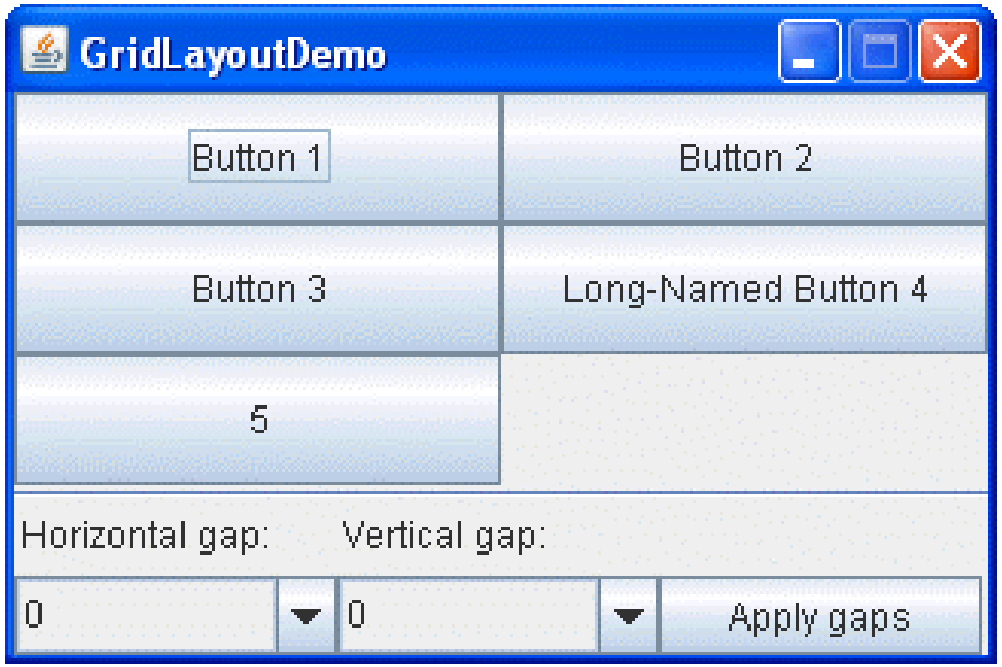
Flow Layout Manager (default)



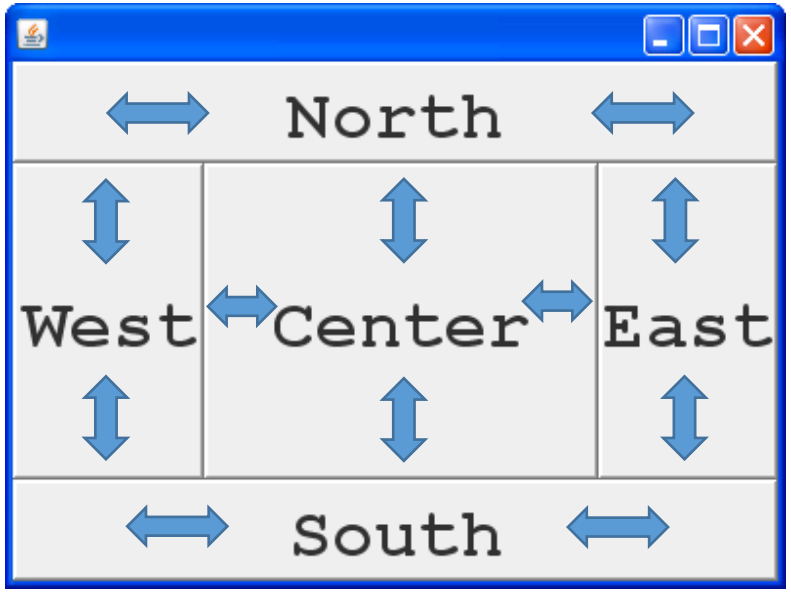
- Like writing or reading English
- Components appear in the order they are added
- Left to right until you get to the right margin
  - Components assume their “natural” (preferred) sizes
  - Default gap between components is 5 pixels
  - How wide is the enclosing panel?
- Then go to a new line

Grid Layout

- Useful when components are in a grid
- Specify rows/cols as argument to constructor
  - 0 indicates “as many as needed”
- All components same size
  - Resized to fit container
- Fills in left to right then top to bottom



Border Layout



- Specify position when adding components
- When resizing… Center expands both x and y
  - North/South expands x, West/East expands y
- By default, there is a 5 pixel gap between components
- Unused positions take no space
- Only one component per position
- Layout hint: panel.add(button,BorderLayout.EAST);
- It's surprising how much this handles!

Component Resizing

- It is possible to set a minimum, preferred, and maximum size on each component

```
yellowLabel.setPreferredSize(new Dimension(200, 180));
yellowLabel.setMinimumSize(new Dimension(100,20));
 yellowLabel.setMaximumSize(new Dimension(600,800));
```

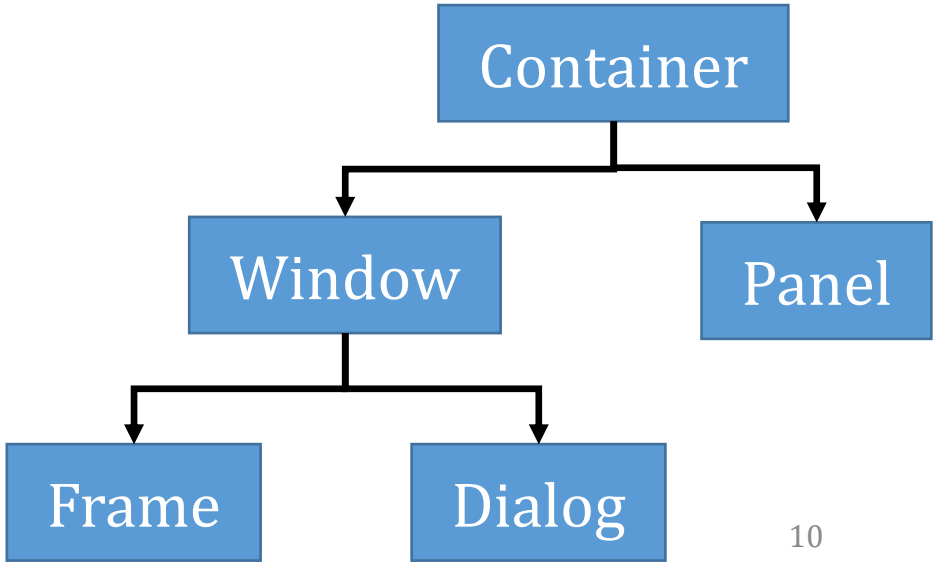
- It would be nice if the layout managers all respected these constraints (they don't – but some, like GridBagLayout, do)

Java Layout Tutorial

- See https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html
- We haven't covered them all… BoxLayout, GridBagLayout, CardLayout, JTabbedPane, SpringLayout, GroupLayout, …

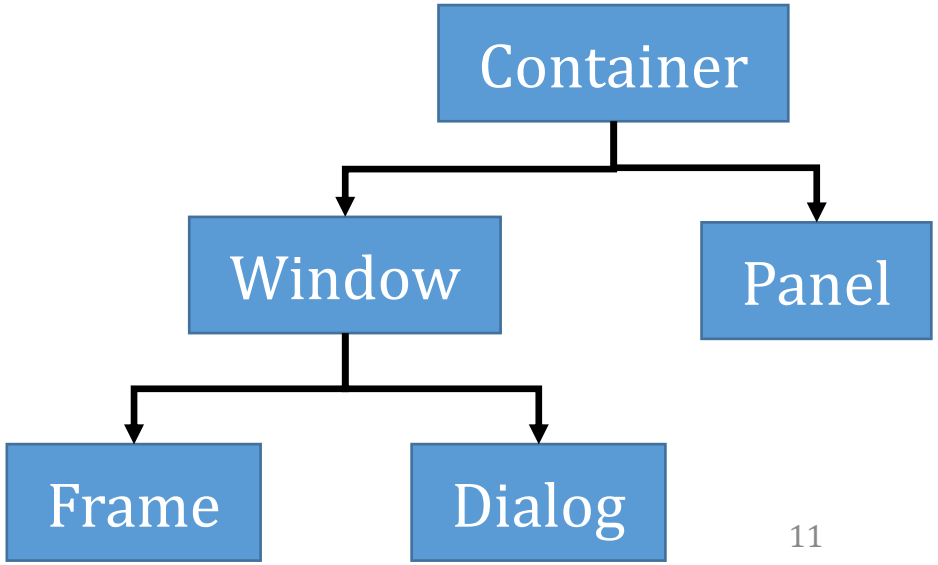
Window vs. Panel

- A "window" is an independent rectangular portion of the display
  - Windows are managed by the X-server
  - Window has a border, title, minimize/maximize/iconize/delete buttons
  - A window can be active or inactive
  - A window can be above or below other windows
- A "panel" is a sub-section of a window
  - Not independently controllable
  - Not independently active or inactive
- Both have components and layout mgrs



Frame vs. Dialog

- A "Frame" is the window for the entire GUI application
  - Program creates Frame, and fills it in
  - Program runs event loop using Frame
  - When frame ends, program usually quits
  - Frame may be inactive – yield to other windows
- A "Dialog" is a special pop-up window
  - To communicate with user
  - Typically, will not yield to other windows
  - Removed when user is done



Popup Dialogs

- There is a JDialog class for creating custom dialogs, but
- Simple popup messages are just…

```
JOptionPane.showMessageDialog(this,"Opcode not recognized.");
```

JOptionPane Static method
Creates Dialog Window: "Opcode not recognized"
with "OK" button to remove dialog and continue