CSE 331

Composite Layouts; Decorators

slides created by Marty Stepp
http://www.cs.washington.edu/331/
Pattern: Composite

objects that can contain their own type
Containers and layout

• Place components in a *container*; add the container to a frame.
  
  **container**: An object that stores components and governs their positions, sizes, and resizing behavior.
Complex layout ... how?

- How would you create a complex layout like this, using only the layout managers shown?
Composite pattern

• **composite**: An object that can be either an individual item or a collection of many items.
  
  ▪ Can be composed of individual items or other composites.
  ▪ Recursive definition: Objects that can hold themselves.
  
  ▪ Often leads to a tree structure of leaves and nodes:
    • `<node>` ::= `<leafnode>` | `<compositenode>`
    • `<compositenode>` ::= `<node>`*

• Examples in Java:
  
  ▪ collections (e.g. a list of lists)
  ▪ GUI layout (containers of containers of components)
**Composite layout**

- **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.

- In the GUI at right:
  - How many containers are there?
  - What layout is used in each?
JPanel

the default container class in Swing

• public JPanel()
  public JPanel(LayoutManager mgr)
  Constructs a panel with the given layout (default = flow layout).

• public void add(Component comp)
  public void add(Component comp, Object info)
  Adds a component to the container, possibly giving extra information about where to place it.

• public void remove(Component comp)

• public void setLayout(LayoutManager mgr)
  Uses the given layout manager to position components.
Flow, Border, Grid layouts

Container panel1 = new JPanel(new FlowLayout());
panel1.add(new JButton("Button 1"));
panel1.add(new JButton("Button 2"));

Container panel2 = new JPanel(new BorderLayout());
panel2.add(new JButton("Button 1 (NORTH)"), BorderLayout.NORTH);

Container panel3 = new JPanel(new GridLayout(3, 2));
panel3.add(new JButton("Button 1"));
panel3.add(new JButton("Button 2"));
Container box1 = Box.createHorizontalBox();
Container box2 = Box.createVerticalBox();

- aligns components in container in a single row or column
- components use preferred sizes and align based on their preferred alignment
  - vertical box is used to get a "vertical flow layout"
Other layouts

- **CardLayout**
  Layers of "cards" stacked on top of each other; one 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)
Composite layout code

Container north = new JPanel(new FlowLayout());
north.add(new JButton("Button 1"));
north.add(new JButton("Button 2"));

Container south = new JPanel(new BorderLayout());
south.add(new JLabel("Southwest"), BorderLayout.WEST);
south.add(new JLabel("Southeast"), BorderLayout.EAST);

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

frame.add(overall);
Pattern: Decorator

objects that "wrap" other objects to add features
JTextField, JTextArea

an input control for typing text values
(field = single line; area = multi-line)

- public JTextField(int columns)
  public JTextArea(int lines, int columns)
  Creates a new field, the given number of letters wide.

- public String getText()
  Returns the text currently in the field.

- public void setText(String text)
  Sets field's text to be the given string.

  What if the text area is too big to fit in the window?
JScrollPane

a container that adds scrollbars around any other component

• public JScrollPane(Component comp)
  Wraps the given component with scrollbars.
  ▪ After constructing the scroll pane, you must add the scroll pane, not the original component, to the onscreen container:

     myContainer.add(new JScrollPane(textarea), BorderLayout.CENTER);
Decorator pattern

- **decorator**: An object that modifies behavior of, or adds features to, another object.
  - Must maintain the common interface of the object it wraps up.
  - Used so that we can add features to an existing simple object without needing to disrupt the interface that client code expects when using the simple object.
  - The object being "decorated" usually does not explicitly know about the decorator.

- Examples in Java:
  - Multilayered input streams adding useful I/O methods
  - Adding scroll bars to GUI controls
Decorator example: I/O

• normal InputStream class has only public int read() method to read one letter at a time

• decorators such as BufferedReader or Scanner add additional functionality to read the stream more easily

```java
// InputStreamReader/BufferedReader decorate InputStream
InputStream in = new FileInputStream("hardcode.txt");
InputStreamReader isr = new InputStreamReader(in);
BufferedReader br = new BufferedReader(isr);

// because of decorator streams, I can read an
// entire line from the file in one call
// (InputStream only provides public int read() )
String wholeLine = br.readLine();
```
Decorator example: GUI

• JScrollPane is a container with scroll bars to which you can add any component to make it scrollable

```java
// JScrollPane decorates GUI components
JTextArea area = new JTextArea(20, 30);
JScrollPane sp = new JScrollPane(area);
contentPane.add(sp);
```

- Components also have a `setBorder` method to add a "decorative" border. Is this another example of the Decorator pattern? Why or why not?
JOptionPane

- `JOptionPane.showMessageDialog(parent, message);`

```java
import javax.swing.*; JOptionPane.showMessageDialog(null, "This candidate is a dog. Invalid vote.");
```

- Advantages:
  - Simple; looks better than console.

- Disadvantages:
  - Created with static methods; not object-oriented.
  - Not powerful (just simple dialog boxes).
More JOptionPane

• `JOptionPane.showMessageDialog(parent, message)`
  ▪ Displays a message and list of choices Yes, No, Cancel.
  ▪ Returns an `int` such as `JOptionPane.YES_OPTION` or `NO_OPTION` to indicate what button was pressed.

• `JOptionPane.showInputDialog(parent, message)`
  ▪ Displays a message and text field for input.
  ▪ Returns the value typed as a String (or `null` if user presses Cancel).