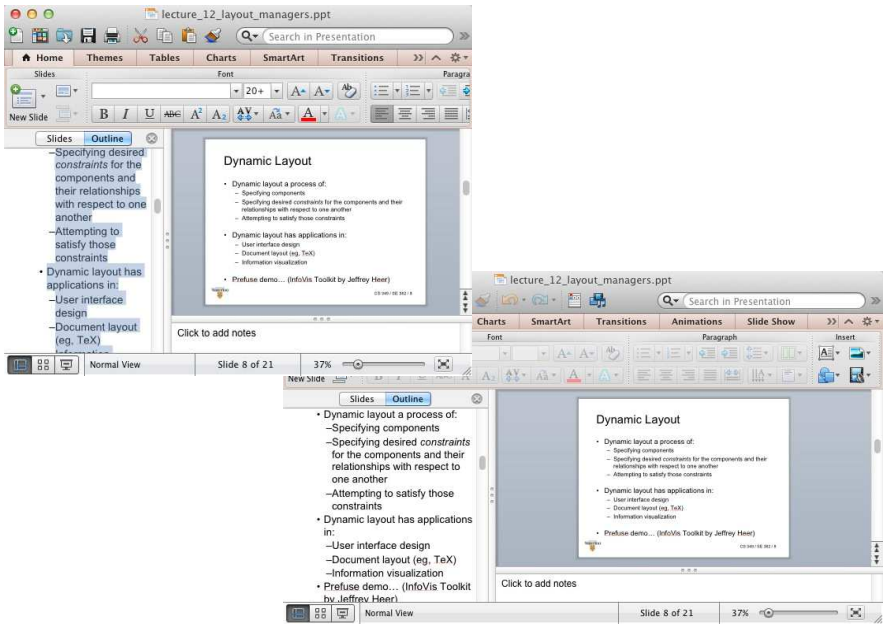
Layout

Interface Layout

- Layout of components can be thought of as two processes:
  - Determining an optimal visual layout (ie, applying principles of good graphic design)
  - Applying algorithms that maintain that desired visual layout through resizes of window
- This lecture focuses on the latter



3

Dynamic Layout

- Windows are dynamic, can be resized
- Through any resize, we wish to:
  - Maintain consistency in interface's presentation
  - Preserve affordances communicated through interface's layout
- Need to dynamically modify allocation of space, locations of objects in interface

4

Dynamic Layout

- Dynamic layout a process of:
  - Specifying components
  - Specifying desired constraints for the components and their relationships with respect to one another
  - Attempting to satisfy those constraints
- Dynamic layout has applications in:
  - User interface design
  - Document layout (eg, TeX)

5

Layout Design Patterns

- Layout in Java makes heavy use of two design patterns:
  - Strategy Pattern
  - Composite Pattern

6

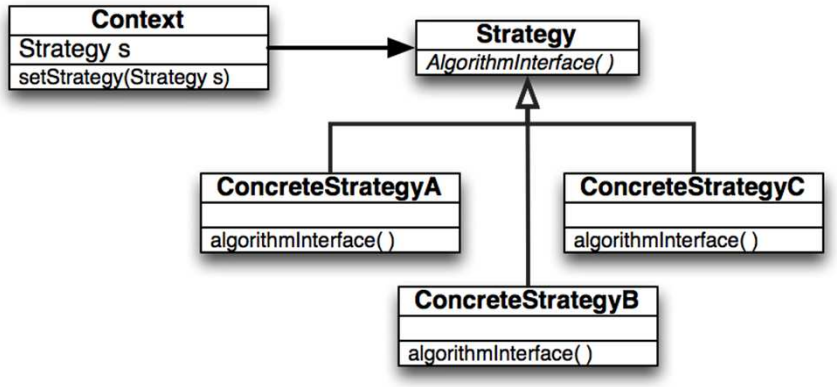
Strategy Design Pattern

- Factors out an algorithm into separate object, allowing a client to dynamically switch algorithms
- Really simple example:
  - quicksort's compare function for sorting any data set
- Other examples?

- A container has a LayoutManager: an object that factors out the algorithm to size & position the container's components
  - eg: container.setLayout(new GridLayout(2, 3));
  - Can vary the LayoutManager independently of container and components

7

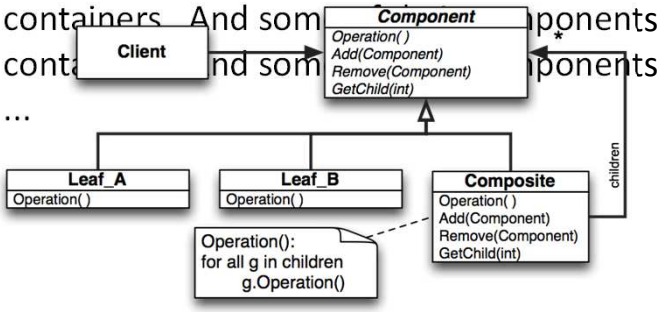
Strategy Design Pattern



8

Composite Design Pattern

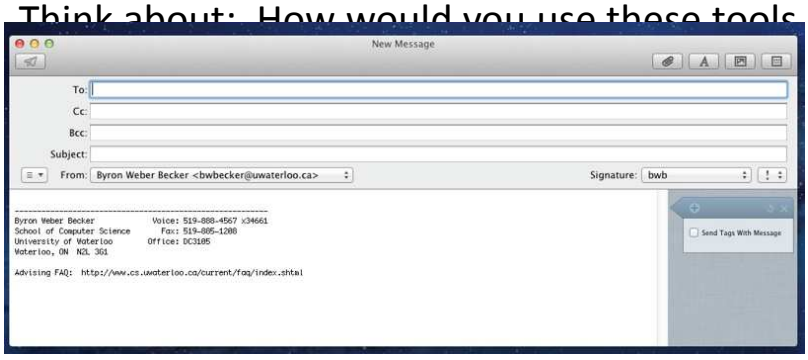
- Containers maintain a collection of components
  - A container is a component
  - So some of the components a container maintains may be containers. And some components may be containers and some components may be ...



9

Java Layout Demo

- LayoutDemo.java
- Available on CS349 Resources page.
- Think about: How would you use these tools



10

General Layout Strategies

- Fixed layout
- Intrinsic size
- Variable intrinsic size
- Struts and springs
- Constraints

11

Fixed Layout

- Components are of a fixed size, position
- In Java, achieved by setting LayoutManager to null
- Where/when is this practical?
- How can it break down even when windows aren't resized?

12

Intrinsic Size Layout

- Query each item for its preferred size
- Grow the component to perfectly contain each item
- A bottom-up approach where top-level component's size completely dependent on its contained components
- Example LayoutManagers in Java that use this strategy
  - BoxLayout, FlowLayout
- Examples of use in interface design?

13

Variable Intrinsic Size Layout

- Layout determined in bottom-up and top-down phases
- Example LayoutManagers in Java
  - GridBagLayout
  - BorderLayout

```
public void doLayout(Rectangle bounds) {
foreach child widget C {
   get preferred size of C
}
decide where each child widget should go
foreach child widget C {
   C.doLayout(new bounds for C);
}
}
```

14

Struts and Springs Layout

- Layout specified by marking aspects of components that are fixed vs. those that can "stretch"
- Strut defines a fixed length (width/height)
  - Specifies invariant relationships in a layout
- Spring defines a space that "pushes" on nearby edges
  - Specifies variable relationships
  - Called "Glue" in Java
- Example LayoutManagers in Java

15

Struts and Springs Uses

- One of the most common strategies, especially in user interface builders
- Provides easily accessible metaphors for people performing layout
- Difficult to layout by hand

16

Constraints-based Layout

- Specify the mathematical relationships between components of the interface.
  - All of the layout managers have constraints to some degree.
  - This is meant to be more general.
- Prefuse takes it to a new level
  - Demo
    - AggregateDemo
    - GraphView
    - Fisheye Menu
    - TreeView

17

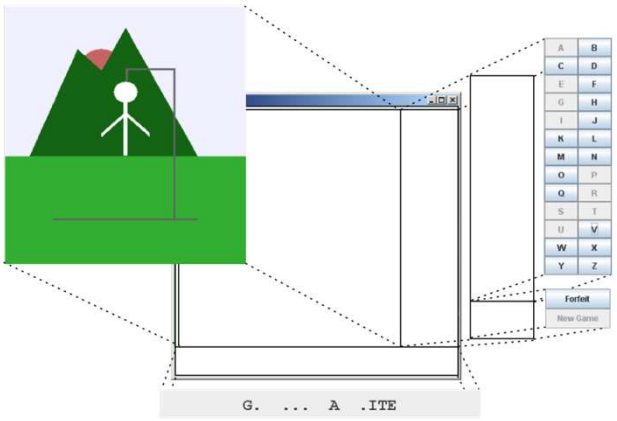
Java Tips and Strategies

- javax.swing.Box has number of useful items that can be used in any layout manager
  - "Glue"
    - Box.createHorizontalGlue()
    - Box.createVerticalGlue()
    - Similar to notion of "springs": Expands to fill space
  - Struts
    - Box.createHorizontalStrut(...)
    - Box.createVerticalStrut(...)
  - Rigid areas
    - Box.createRigidArea(...)

18

Tips and Strategies

- Break up the UI recursively with panels that contain panels.
- Cluster components into panels



19

Tips and Strategies

- Define your own layout manager if necessary
  - See FormLayout.java in the Model-View-Controller sample code.

```
public interface LayoutManager
{      void addLayoutComponent(String name, Component
comp);
       void removeLayoutComponent(Component comp);
       Dimension preferredLayoutSize(Container parent);
       Dimension minimumLayoutSize(Container parent);
       void layoutContainer(Container parent);
}
// LayoutManager2 has methods for specifying constraints
```

20