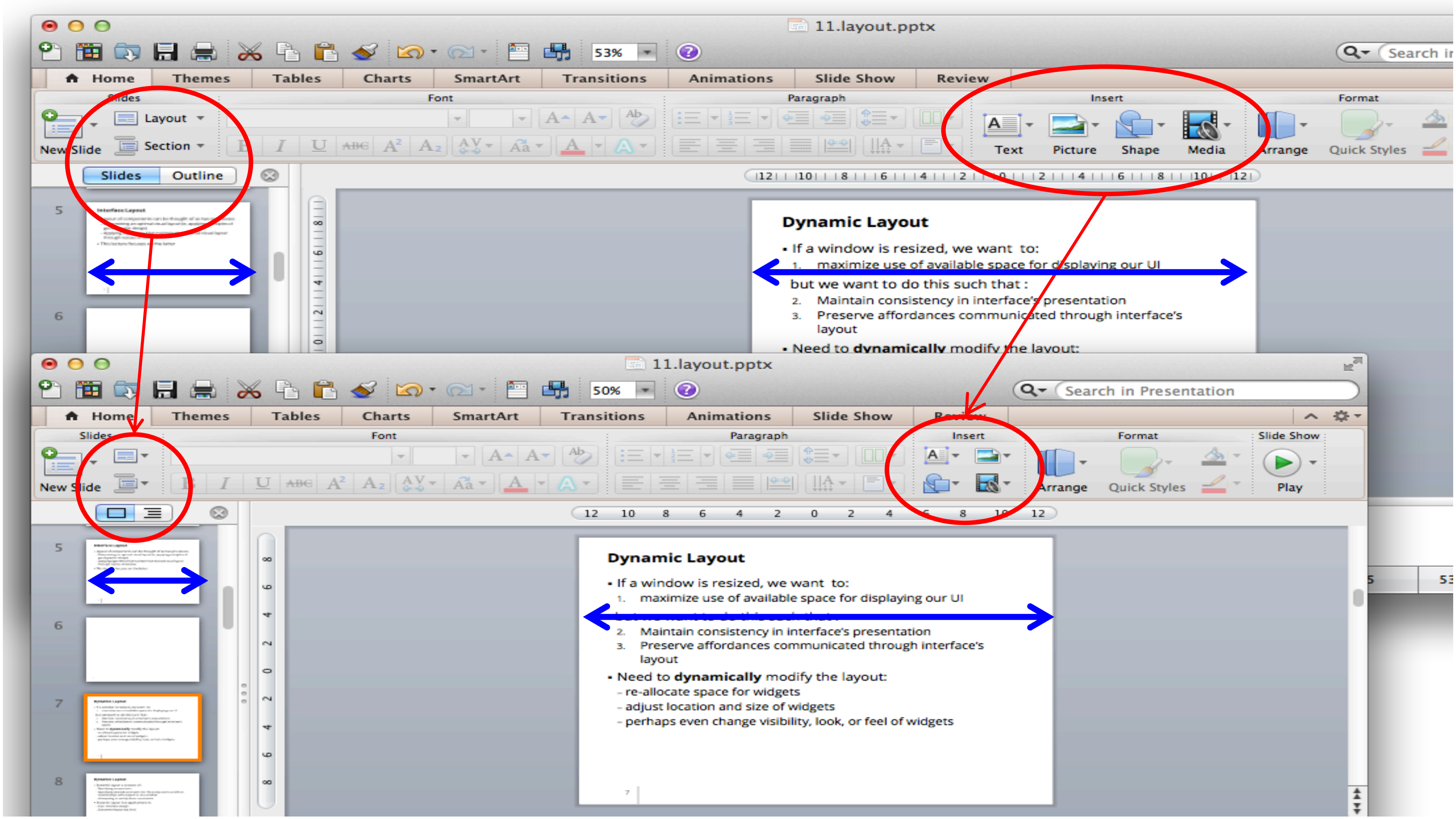
Layout

Dynamic layout

Swing and Layout Managers

Layout strategies



Dynamic Layout

- If a window is resized, we want to:
  1. maximize use of available space for displaying our UI

but we want to do this such that :

  2. Maintain consistency in interface's presentation
  3. Preserve affordances communicated through interface's layout

- Need to **dynamically** modify the layout:

Dynamic Layout

- If a window is resized, we want to:
  1. maximize use of available space for displaying our UI

but we want to do this such that :

  2. Maintain consistency in interface's presentation
  3. Preserve affordances communicated through interface's layout

- Need to **dynamically** modify the layout:
  - re-allocate space for widgets
  - adjust location and size of widgets
  - perhaps even change visibility, look, or feel of widgets

Two Interface Layout Tasks

1. Designing a spatial layout of widgets in a container
2. Adjusting that spatial layout when container is resized
   - Both can be done by hand (i.e. graphic design) or automatically (i.e. with algorithms).

- Spatial layout is one component of visual design, so:
  - should use/maintain Gestalt Principles
  - should use/maintain grouping, hierarchy, relationships, balance to achieve organization and structure
- We’ll revisit this later in the course.

Dynamic Layout

- If a window is resized, we want to:
  - maximize use of available space for displaying widgets

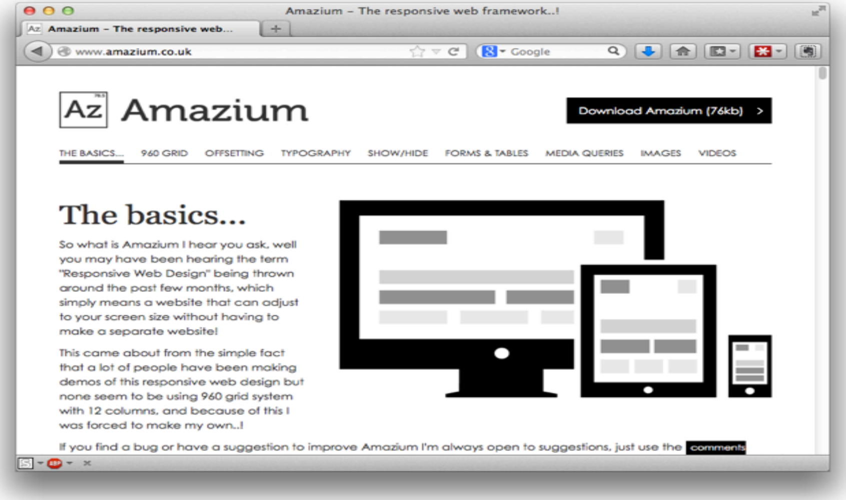
- but we want to do this such that :
  - maintain consistency with spatial layout
  - preserve visual quality of spatial layout

- Need to dynamically modify the layout:
  - re-allocate space for widgets
  - adjust location and size of widgets
  - perhaps even change visibility, look, and/or feel of widgets

Adaptive/Responsive Layout

- Changing layout to adapt/respond to different devices
  - e.g. same web page with layouts for desktop, tablet, smartphone
- Often goes beyond spatial layout to swapping widgets
- Dynamic layout a special case of adaptive/responsive layout

http://www.amazium.co.uk/

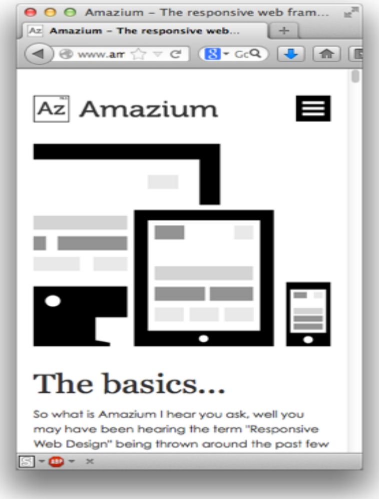


The basics...

So what is Amazium I hear you ask, well you may have been hearing the term "Responsive Web Design" being thrown around the past few months, which simply means a website that can adjust to your screen size without having to make a separate website!

This came about from the simple fact that a lot of people have been making demos of this responsive web design but none seem to be using 960 grid system with 12 columns, and because of this I was forced to make my own..!

If you find a bug or have a suggestion to improve Amazium I'm always open to suggestions, just use the comments


Amazium – The responsive web fram...
Amazium – The responsive web...
www.am
Gc
Az Amazium

The basics...

So what is Amazium I hear you ask, well you may have been hearing the term "Responsive Web Design" being thrown around the past few

Widget Size and Position

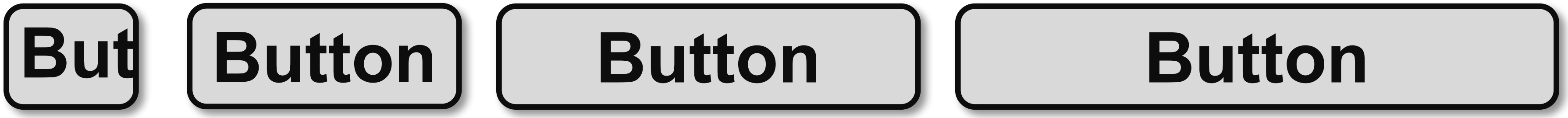
To make a layout dynamic, widgets need to be “flexible”

- – x,y position may be changed
- – width and height may be changed

However, these changes can be constrained

- Widgets give the layout algorithm constraints on position
  - – e.g. must be anchored on the left side of the window
- Widgets give the layout algorithm a range of sizes:

minimum size < preferred size < maximum size



Layout Managers

A Layout Manager provides a layout algorithm to size and position child widgets.

Java's Swing package provides a number of layout managers: `Grid, Box, Border, Flow, GridBag`, etc.

A widget can set the most appropriate layout strategy:

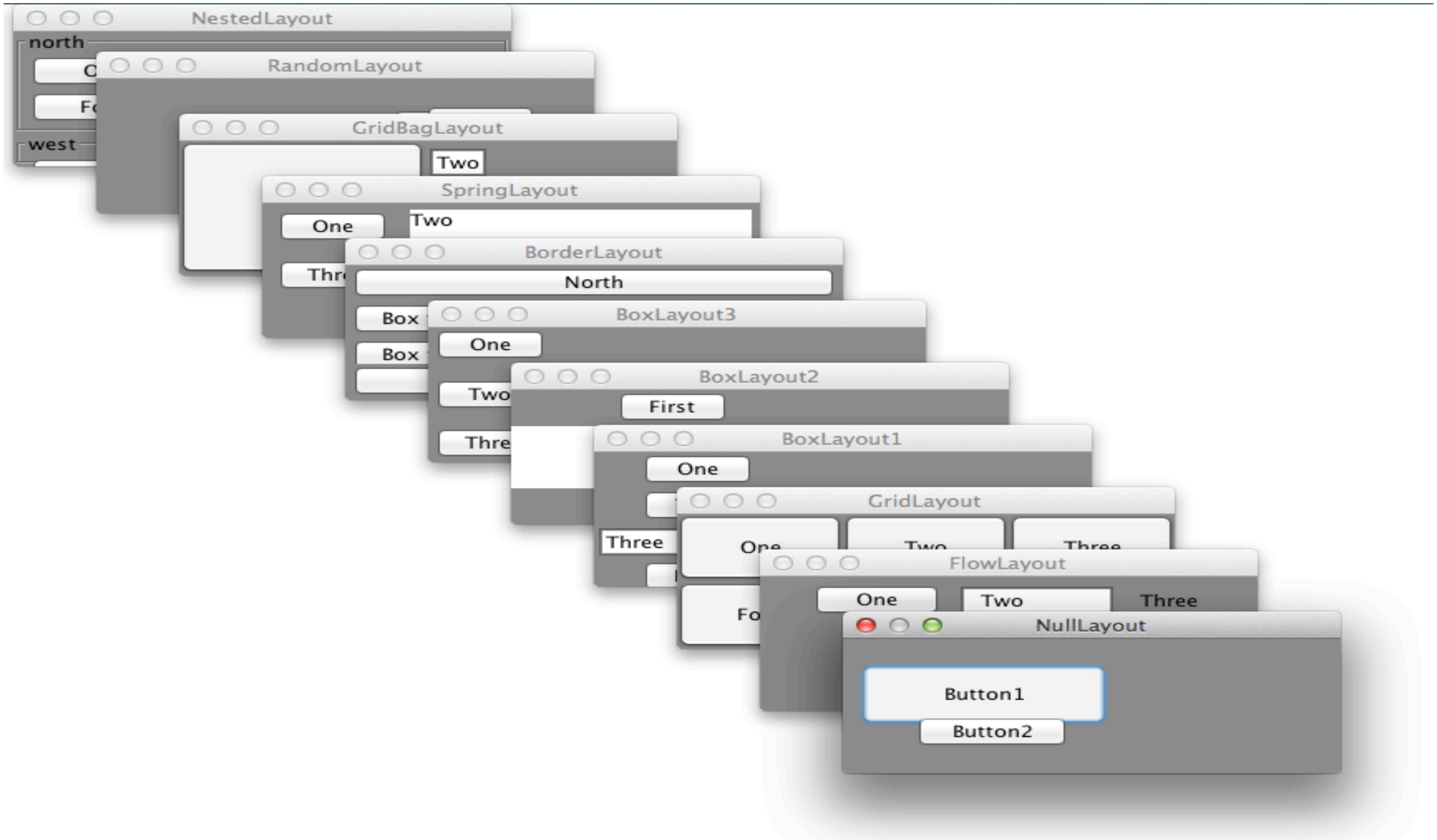
```
container.setLayout(new GridLayout(2, 3));
```

Most useful for container widgets like JPanel

Layout Manager Attributes

- Does it respect a widget's preferred/min/max size?
  - Always ignored?
  - Always respected, even if parts of a widget extend off the edge?
  - Respected in some dimensions but not others?
- How does it handle extra space?
  - Add extra space around widgets?
  - Give it equally to all widgets?
  - Give it unequally to widgets?
- Do widgets require additional constraints?
  - Where in the layout manager?
  - Alignment?
  - Share of additional space?

Code Demo: LayoutDemo.java



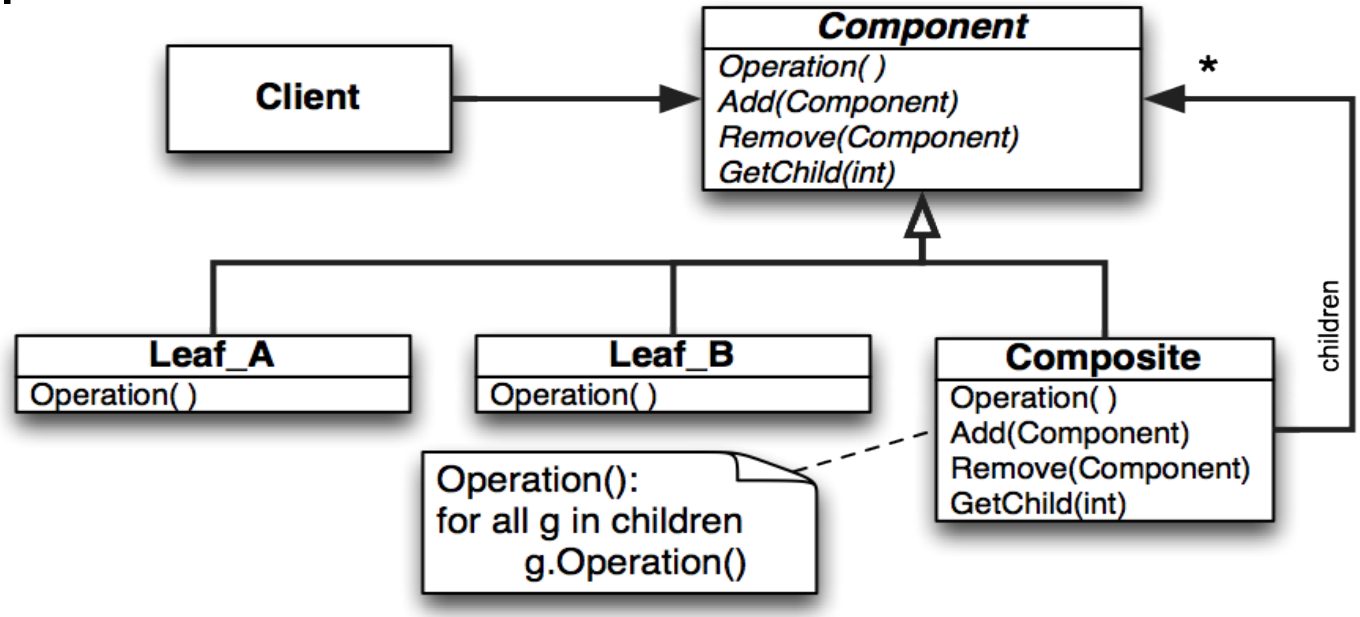
Layout Design Patterns

Layout in Java makes heavy use of two design patterns:

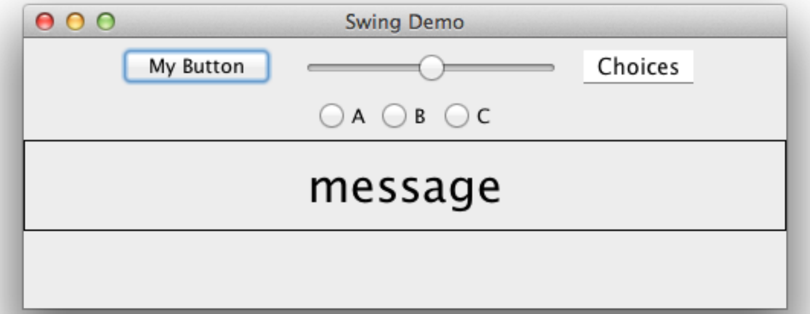
- Composite Pattern
- Strategy Pattern

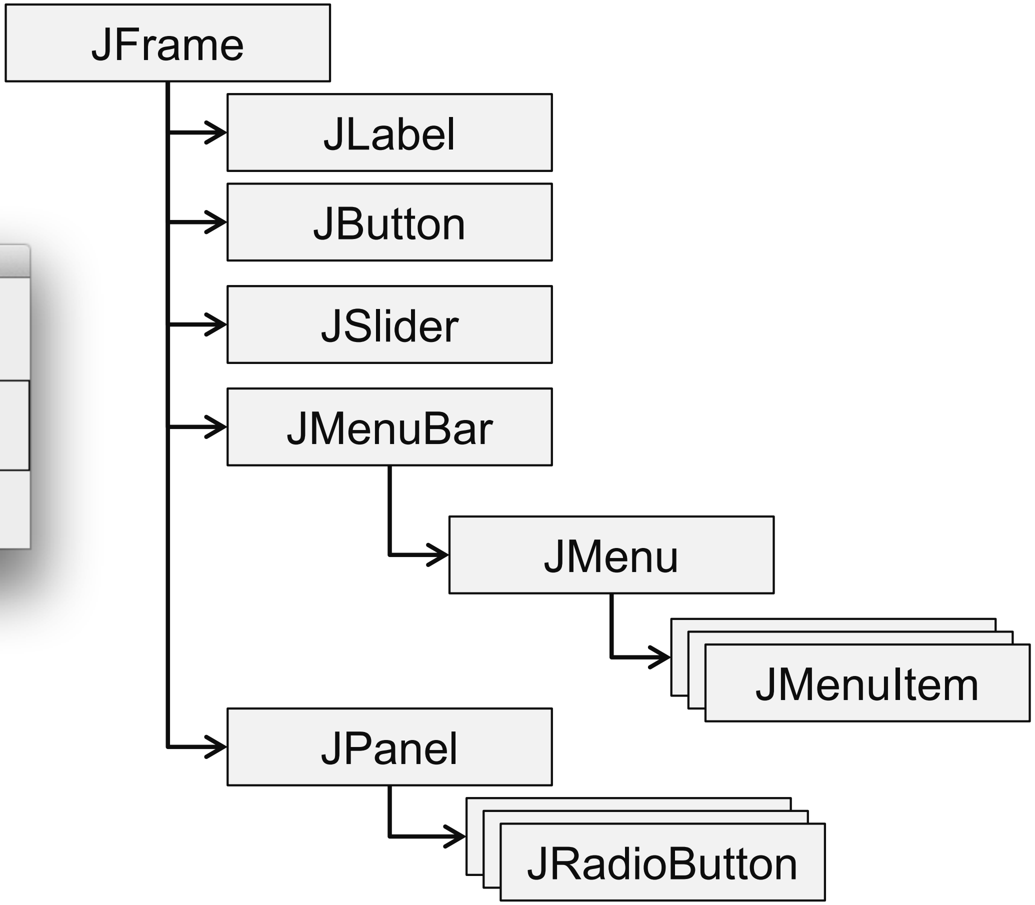
Composite Design Pattern

The composite pattern specifies that a group of objects are to be treated in the same way as a single instance of an object. The intent of a composite is to “compose” objects into tree structures to represent part-whole hierarchies. Implementing the composite pattern lets clients treat individual objects and compositions uniformly.



Composite Pattern with Swing





JFrame

JLabel

JButton

JSlider

JMenuBar

JMenu

JMenuItem

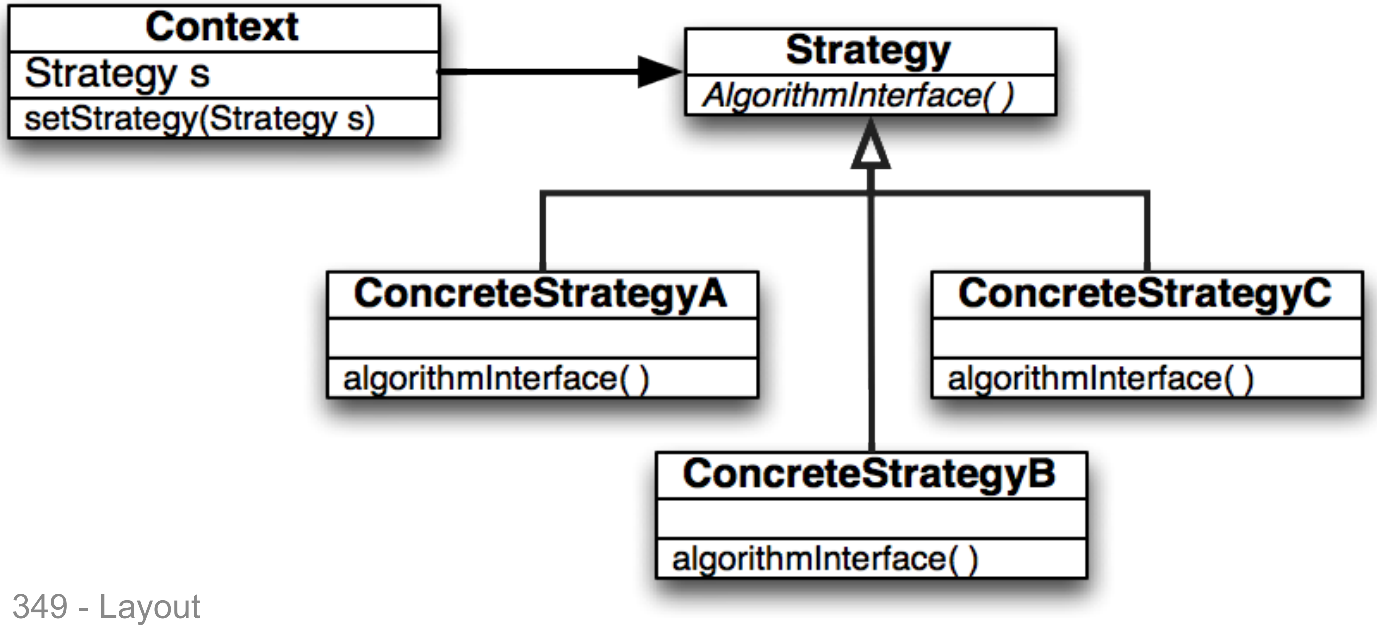
JPanel

JRadioButton

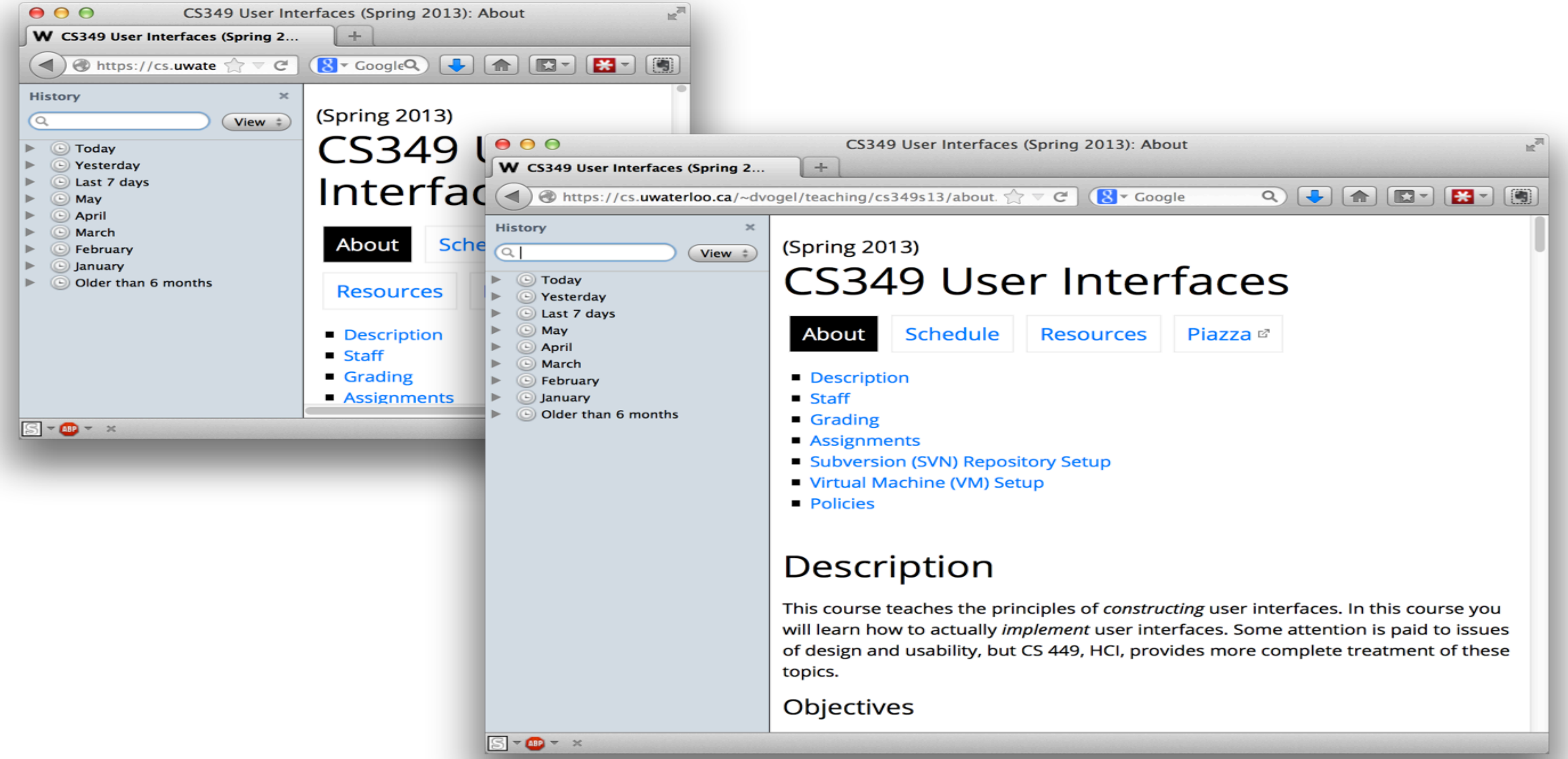
Strategy Design Pattern

Factors out an algorithm into separate object, allowing a client to dynamically switch algorithms

- e.g. Java Comparator “strategy” for a Collection “context”, switching a game’s move selection algorithm from “easy” to “hard”, text formatter for textboxes, etc.



How to implement this layout?



(Spring 2013)

CS349 User Interfaces

- Description
- Staff
- Grading
- Assignments
- Subversion (SVN) Repository Setup
- Virtual Machine (VM) Setup
- Policies

Description

This course teaches the principles of *constructing* user interfaces. In this course you will learn how to actually *implement* user interfaces. Some attention is paid to issues of design and usability, but CS 449, HCI, provides more complete treatment of these topics.

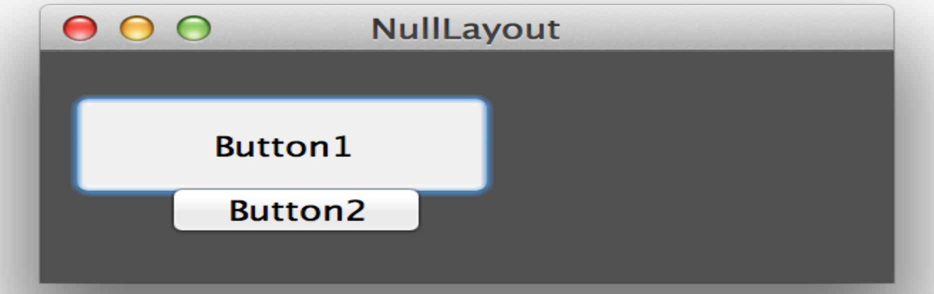
Objectives

General Layout Strategies

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

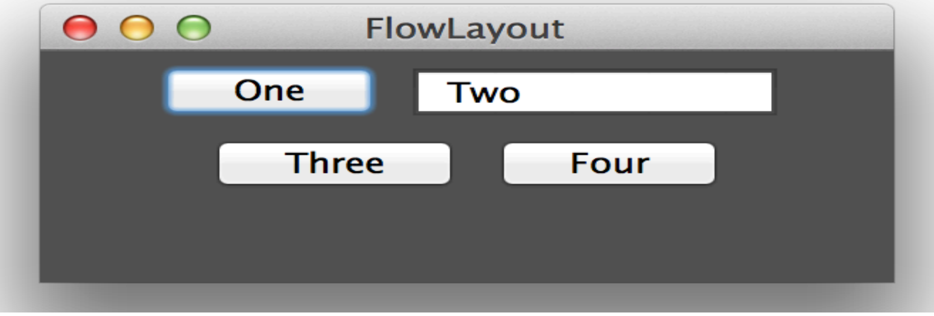
Fixed Layout

- Widgets have a fixed size, fixed 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?



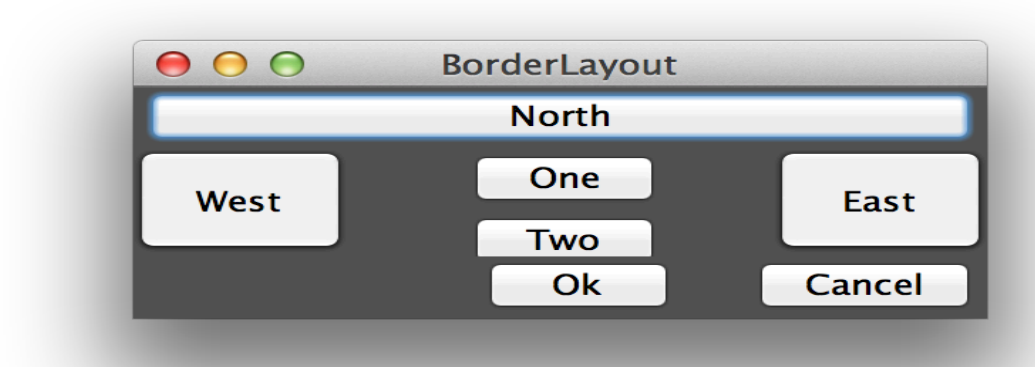
Intrinsic Size Layout

- A bottom-up approach where top-level widget's size is completely dependent on its contained widgets
- Single pass algorithm
  - Query each child widget for its preferred size
  - Adjust the parent widget to perfectly contain each item
- Example LayoutManagers in Java that use this strategy
  - BoxLayout, FlowLayout
- Examples of use in interface design?
- Special needs?



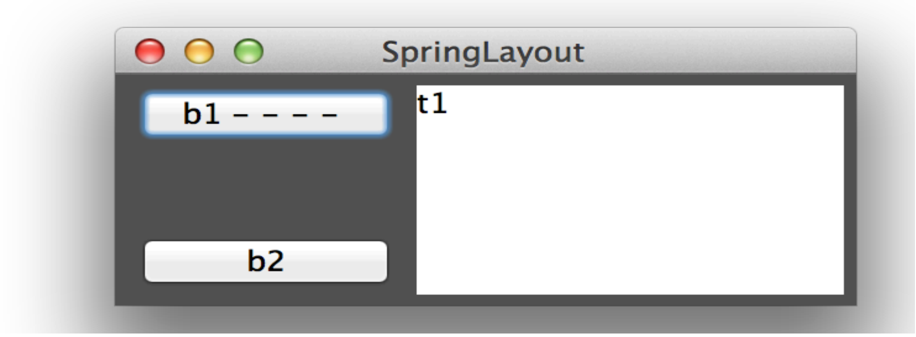
Variable Intrinsic Size Layout

- Set each child widget’s size and location based on the child’s preferences and the parent’s algorithm
- Layout determined in two-passes (bottom-up, top-down)
  1. Get each child widget’s preferred size (includes recursively asking all of its children for their preferred size…)
  2. Decide on a layout that satisfies everyone’s preferences, then iterate through each child, and set it’s layout (size/position)
- Example LayoutManagers in Java that use this strategy
  – GridBagLayout
  – BorderLayout



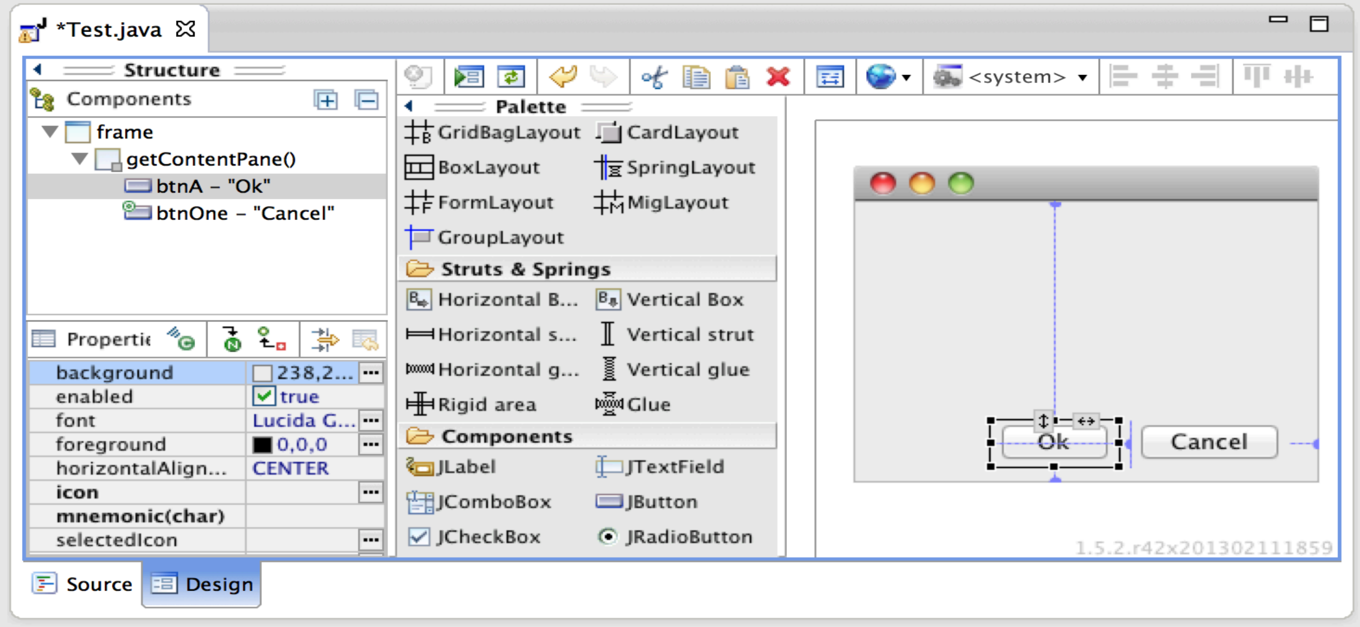
Struts and Springs Layout

- Layout specified by marking aspects of widgets that are fixed vs. those that can “stretch”
- Strut is a fixed space (width/height)
  - Specifies invariant relationships in a layout
- Spring “stretches” to fill space (or expand widget size)
  - Specifies variable relationships
  - Springs are called “glue” in Java
- Can add more general constraints too
  - e.g. widget must be EAST of another widget
- Example LayoutManagers in Java
  - SpringLayout,
    BoxLayout (restricted form)



Struts and Springs in GUI Design Tools

- Very common, especially in Interactive GUI design tools
  - Can be more difficult to hand code
- Good metaphors for people performing layout



- frame
  - getContentPane()
    - btnA – "Ok"
    - btnOne – "Cancel"

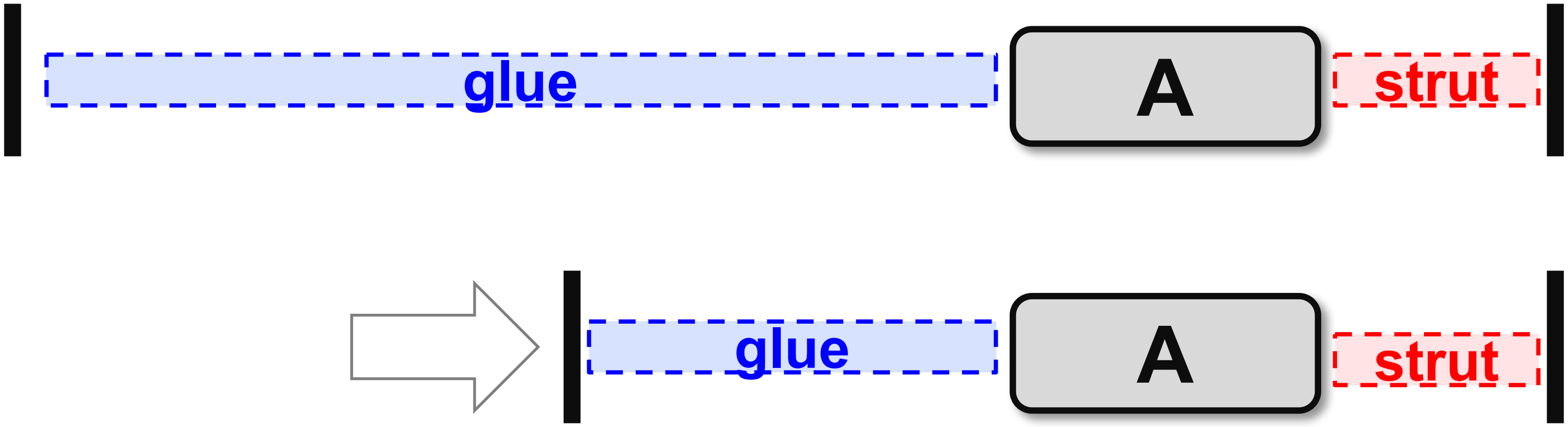
Propertie

background	238,2...
enabled	☑ true
font	Lucida G...
foreground	0,0,0
horizontalAlign...	CENTER
icon	
mnemonic(char)	
selectedIcon	

Google WindowBuilder Eclipse Plug-in

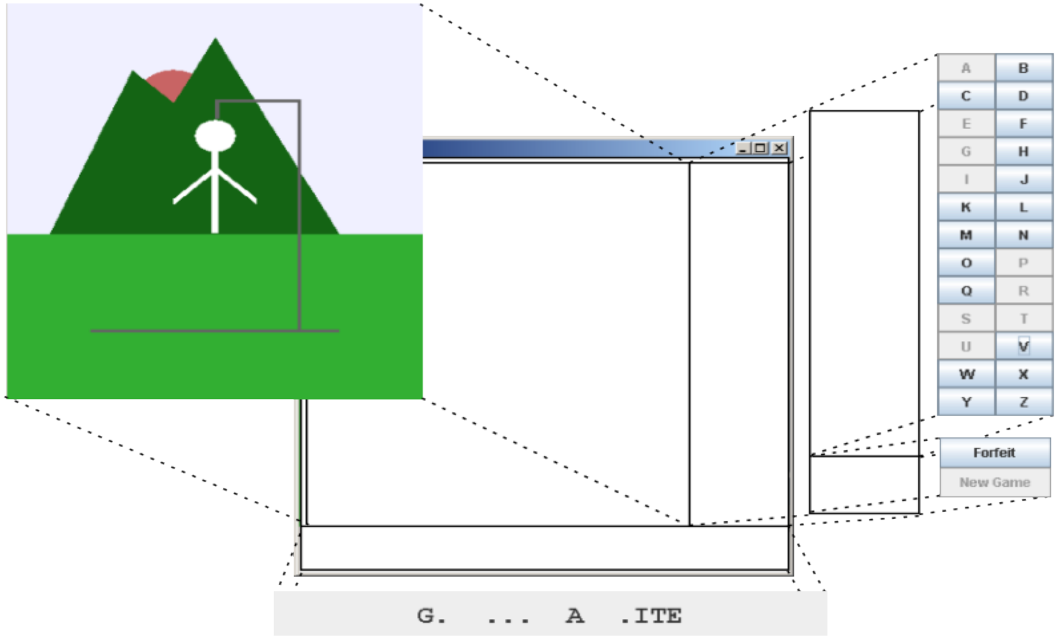
Java Tips and Strategies

- **javax.swing.Box** has useful widgets for any layout manager
  - Glue to expand/contract to fill space (i.e. “Springs”)
    - Box.createHorizontalGlue(), Box.createVerticalGlue()
  - Rigid Areas and Struts to occupy space
    - Box.createHorizontalStrut(...), Box.createVerticalStrut(...)
    - Box.createRigidArea(...)



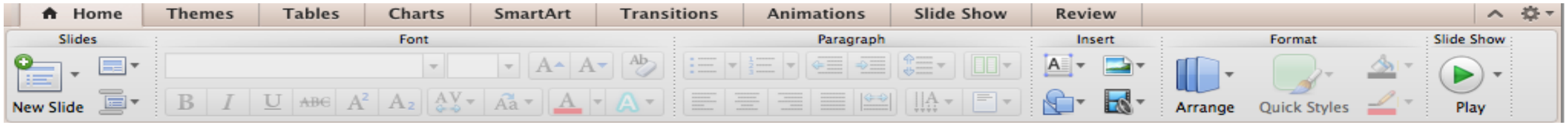
Tips and Strategies

- Break up the UI recursively with panels that contain panels.
- Cluster components into panels based on layout needs
- Provide a layout manager for each panel
- Consider making each panel into a view (see MVC lecture)



Custom Layout Managers

- Creating Ribbon?



- Can’t push it quite far enough with standard Java layouts
  - Need custom layout manager …
  - See RandomLayout in sample code for an example

```
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);
}
```

Creating UIs Using XML

- Many programming languages use XML to create UIs
  - It enables you to better separate the presentation of your application from the code that controls its behavior.
  - Your UI descriptions are external to your application code, which means that you can modify or adapt it without having to modify your source code and recompile.
  - Declaring the layout in XML makes it easier to visualize the structure of your UI, so it's easier to debug problems.
- Problem I see
  - Android context needs to be considered (config doesn’t change)
  - Many xml-based layouts use absolute positioning, relative positioning, etc.
  - A lot of burden on programmer to maintain …

Summary

- Dynamic layout is required to adjust to different window sizes at runtime.
  - Toolkits support fixed layout, or dynamic/algorithmic approaches.
- Layout managers provide different pluggable strategies
- The key to managing a dynamic layout is determining the appropriate strategy.
  - Constraints help the layout meet your design goals.