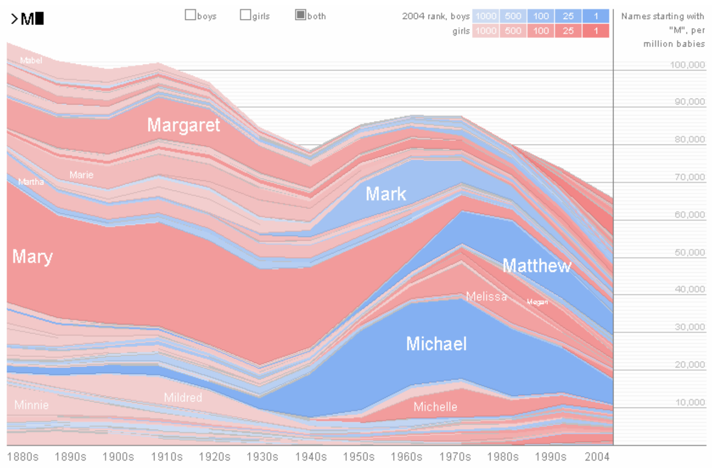
Lecture 12: GUI Builders

UI Hall of Fame or Shame?



Suggested by Casey Dugan

This web site is the Baby Name Wizard's Name Voyager (www.babynamewizard.com).

It's a great example of direct manipulation. The baby names have a continuous visual representation, which I can control by physical actions (clicking on a colored line). The text entry field is an incremental search, so it has rapid, incremental, and reversible effects.

Alas, some actions do *not* have easily reversible effects. Clicking on a name, like Michael, can't be reversed by another click or single keypress; instead, I have to backspace to erase the whole word "Michael" in order to get back to where I was before. That makes it much harder to explore, and makes me (as a user) unwilling to click on something, because the cost of reversing the click isn't worth the benefit I get from zooming in on one name.

Another unfortunate limitation is that I can't zoom or filter the time line. Knowing how popular Mary was in the 1880's is historically interesting, but if I'm trying to choose a name for my baby, the last 10 or even 5 years may be much more important to me, so I'd want to use as much more length of the y axis position for that range.

Another issue is that you can't easily compare names that don't share a common prefix – but comparing names is very important to new parents. This may be a failure of task analysis.

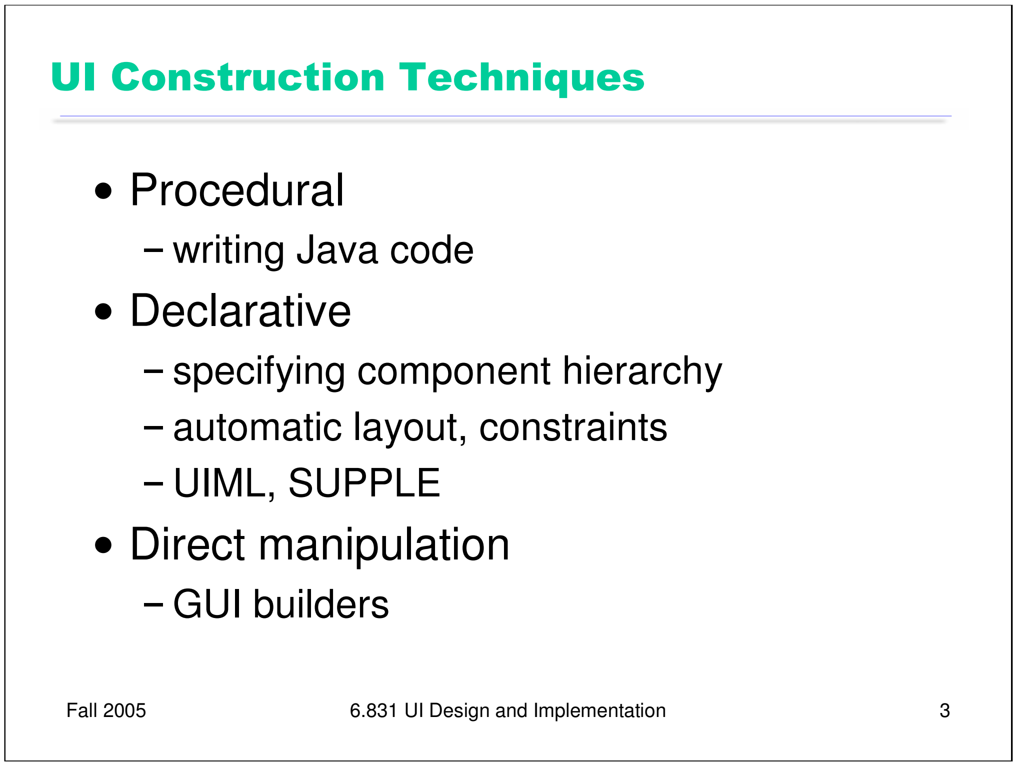
What visual variables are used here?

- **hue** for gender (a nominal variable)

- **saturation** for popularity rank (an ordinal variable)

- **size** (vertically) for frequency (a quantitative variable)

- **position** (horizontally) for time (a quantitative variable)

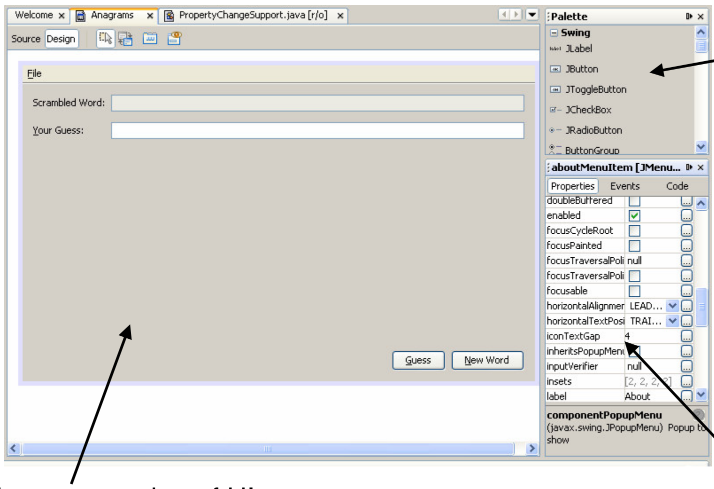


UI Construction Techniques

- Procedural
  - writing Java code
- Declarative
  - specifying component hierarchy
  - automatic layout, constraints
  - UIML, SUPPLE
- Direct manipulation
  - GUI builders

Fall 2005 6.831 UI Design and Implementation 3

GUI Builders



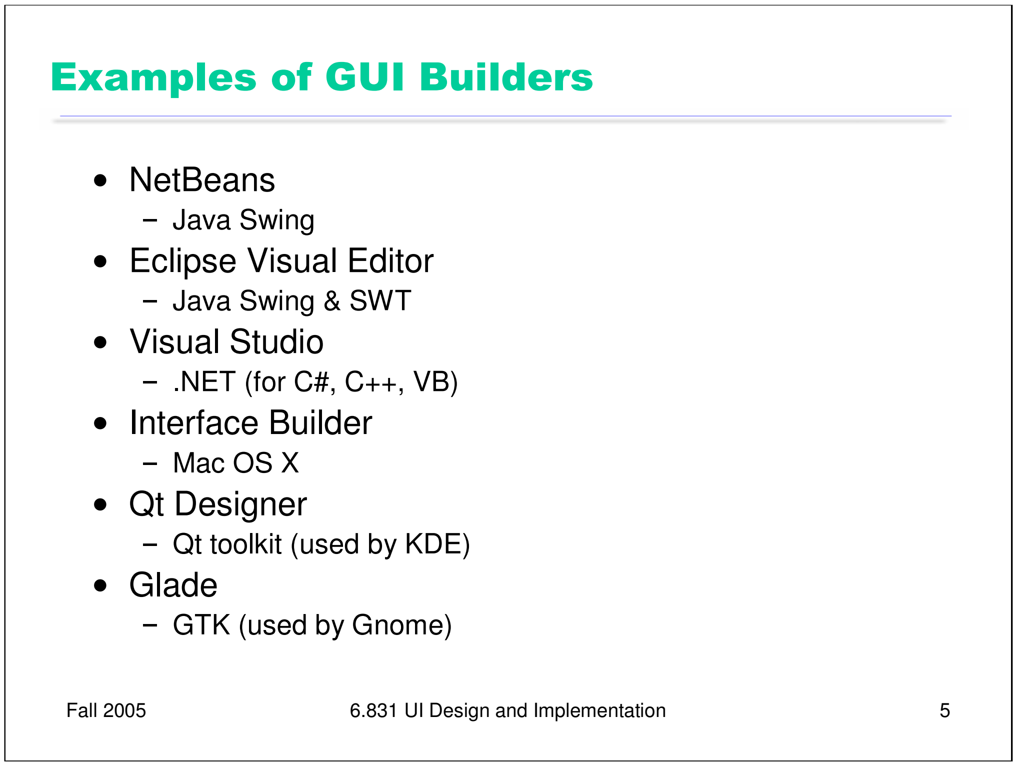
doubleBuffered	☐
enabled	✓
focusCycleRoot	☐
focusPainted	☐
focusTraversalPoli	null
focusTraversalPoli	☐
focusable	☐
horizontalAlignmer	LEAD...
horizontalTextPosi	TRAI...
iconTextGap	4
inheritsPopupMenu	☐
inputVerifier	null
insets	[2, 2, 2, 2]
label	About

componentPopupMenu
(javax.swing.JPopupMenu) Popup to show

A GUI builder basically applies the principles of direct manipulation to user interface design itself.

You see a continuous visual representation of your interface (on the left).

You can change it using physical actions (moving & resizing widgets, dragging and dropping widgets from a palette, click-to-edit labels and buttons, editable property sheets).



Examples of GUI Builders

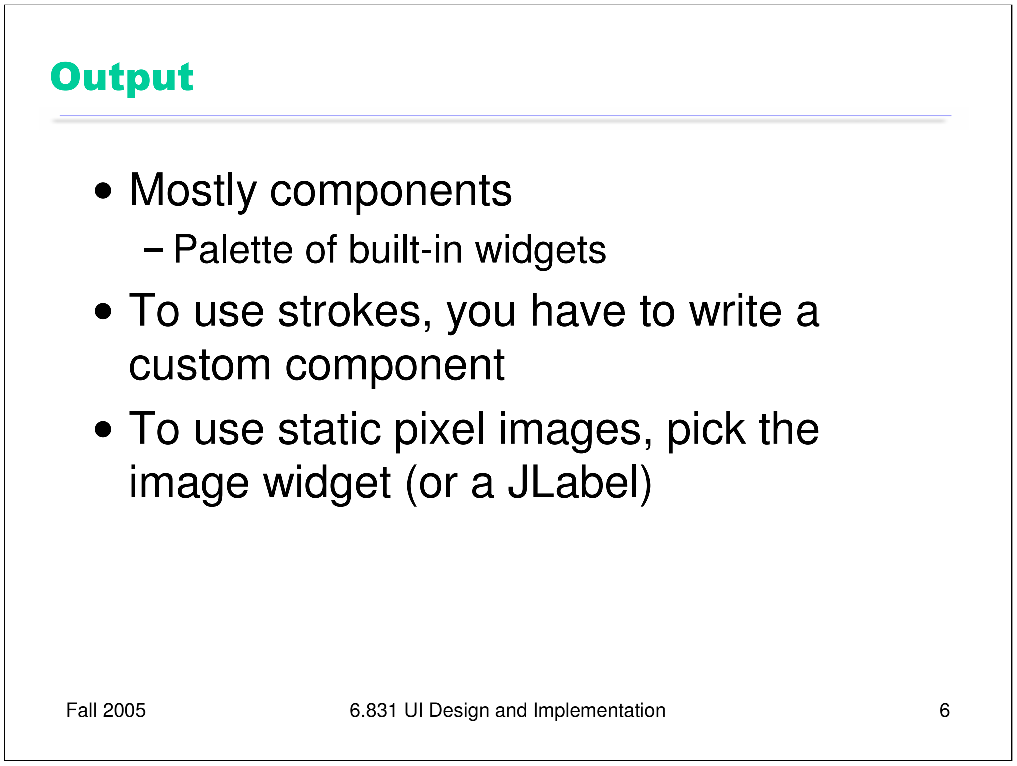
- NetBeans
  - Java Swing
- Eclipse Visual Editor
  - Java Swing & SWT
- Visual Studio
  - .NET (for C#, C++, VB)
- Interface Builder
  - Mac OS X
- Qt Designer
  - Qt toolkit (used by KDE)
- Glade
  - GTK (used by Gnome)

Here are some of the most well-known and mature GUI builders. If you're using Java, the first two are the most interesting:

NetBeans has what is probably the most mature and sophisticated GUI builder for Java Swing apps. You can download it from www.netbeans.org. NetBeans is actually a full IDE, with editors and debuggers, comparable to Eclipse.

Eclipse also has a GUI builder, called Visual Editor, which is available as a separate plugin, not installed by default. To get it: (1) first install Eclipse itself (www.eclipse.org); (2) go to Help >> Software Updates >> Find & Install; (3) choose "Search for new features to install"; (4) choose "Eclipse.org update site"; (5) once it retrieves all the features, open up Eclipse.org update site >> VE >> Visual Editor, and check it; (6) click Select Required to get all the features Visual Editor depends on; (7) click through the rest of the wizard to install it.

Even after you've installed Visual Editor, it's hard to find how to start it. Go to File >> New >> Other… >> Java, and you'll see an option "Visual Class". This will create a new Java class and edit it with Visual Editor.



Output

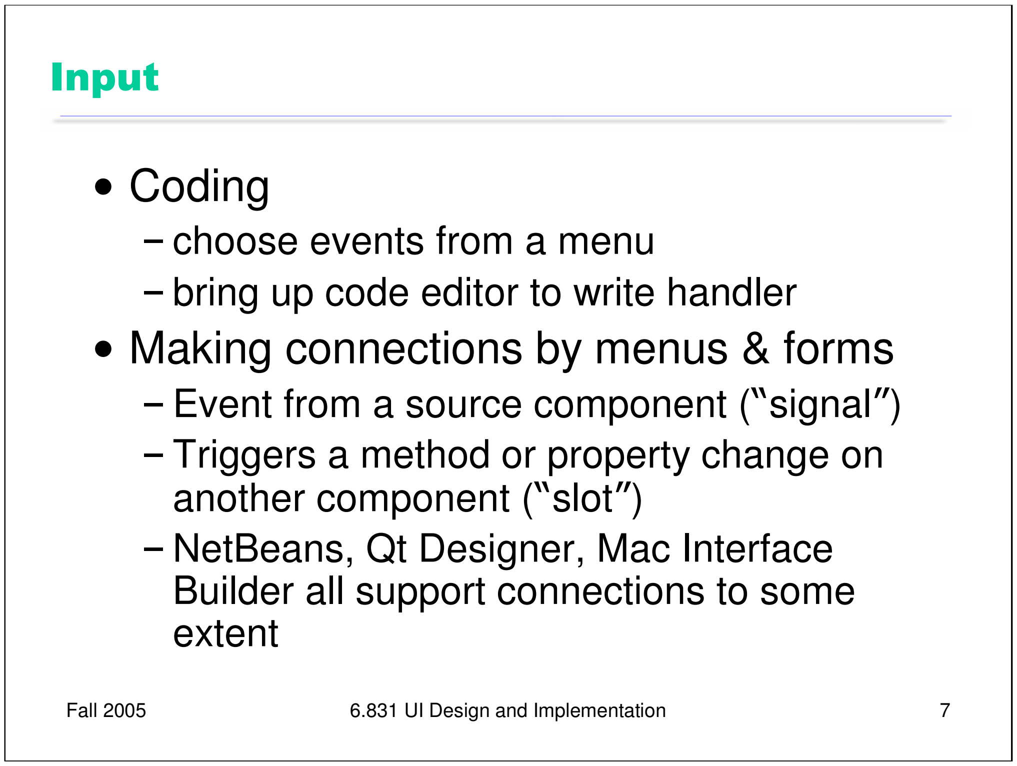
- Mostly components
  - Palette of built-in widgets
- To use strokes, you have to write a custom component
- To use static pixel images, pick the image widget (or a JLabel)

Fall 2005 6.831 UI Design and Implementation 6

GUI builders mainly support the component model for output. They have a palette of standard widgets (buttons, labels, checkboxes, etc).

You can usually add your own custom components to this palette, which is how you can integrate the stroke and pixel models. But custom components must follow some conventions to be fully usable in GUI builder – in Java, your component must be a Java “bean”. we’ll discuss this in more detail a little later.

Another way to use pixel model output is to use a image widget. in Java, JLabel can serve that role, if you leave its text label blank and set an Icon into it (an icon doesn’t have to be small!)



Input

- Coding
  - choose events from a menu
  - bring up code editor to write handler
- Making connections by menus & forms
  - Event from a source component (“signal”)
  - Triggers a method or property change on another component (“slot”)
  - NetBeans, Qt Designer, Mac Interface Builder all support connections to some extent

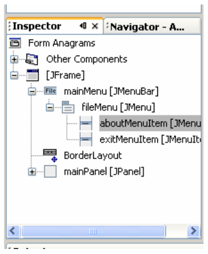
Fall 2005 6.831 UI Design and Implementation 7

For the coding example, we used Eclipse to create an event handler for the File/Exit menu item’s actionPerformed event, and then filled in the body of the handler with System.exit(0).

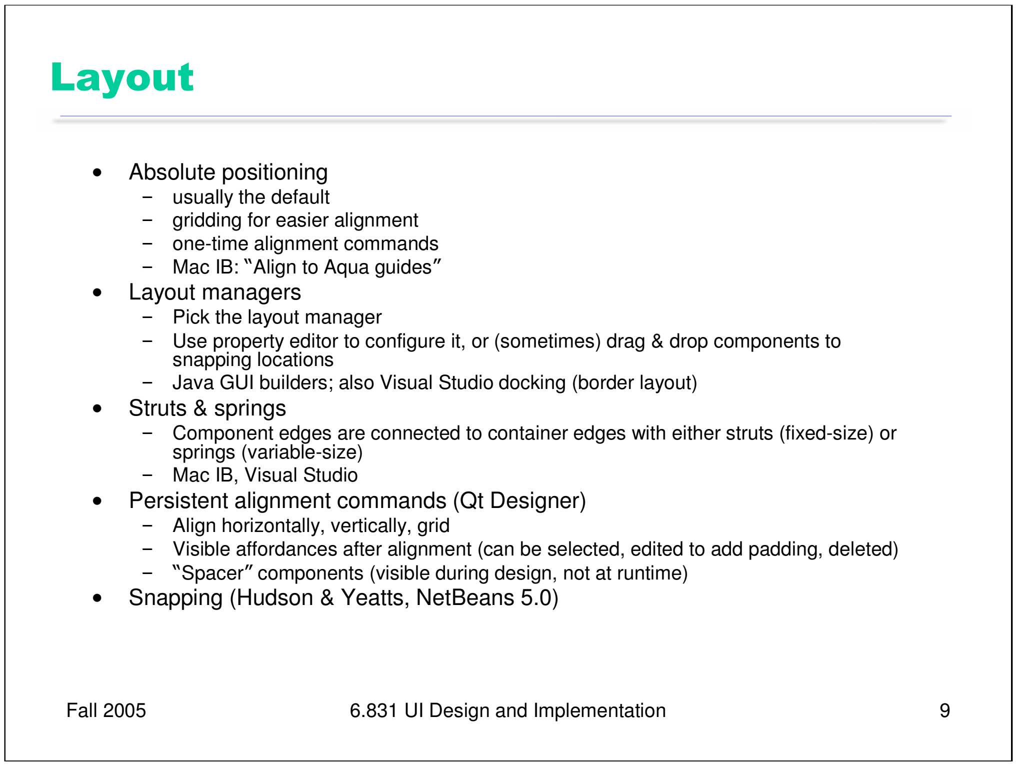
For the connections example, we used NetBeans to connect a textbox’s actionPerformed event to the OK button’s doClick() method.

Component Hierarchy

- Sometimes visually represented
  - NetBeans, Eclipse VE



- Sometimes implicit
  - Dragging a component into a container's bounding box makes it a child
  - Bring to Front / Send to Back control sibling order
  - Visual Studio, Qt Designer



Layout

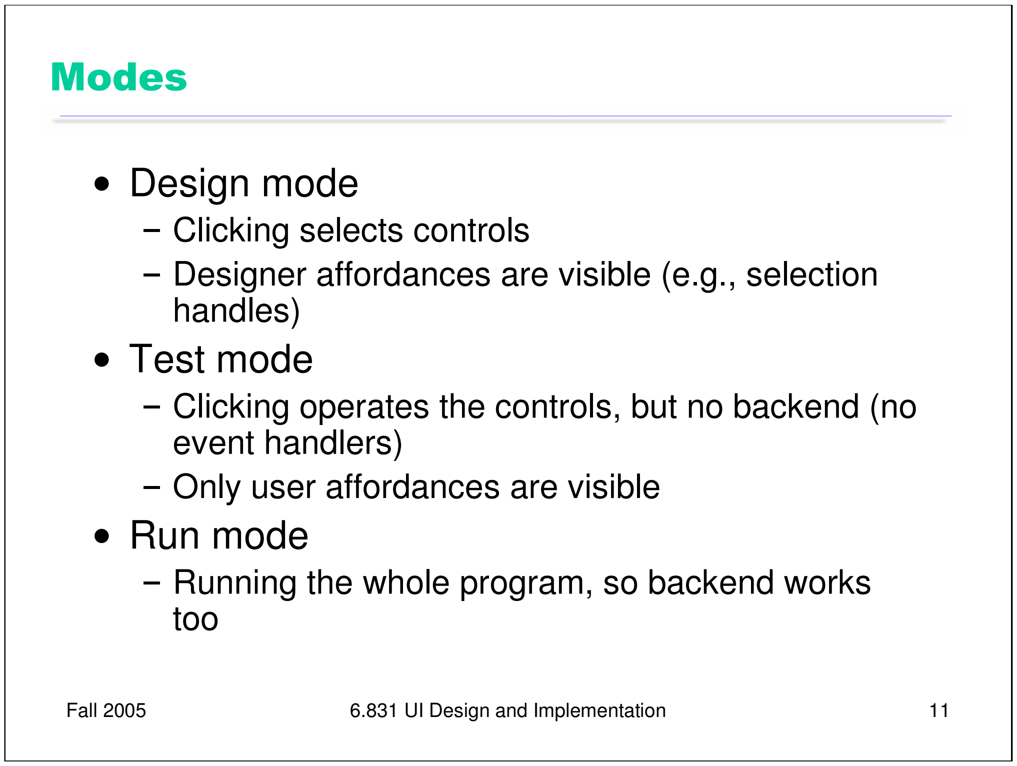
- Absolute positioning
  - usually the default
  - gridding for easier alignment
  - one-time alignment commands
  - Mac IB: "Align to Aqua guides"
- Layout managers
  - Pick the layout manager
  - Use property editor to configure it, or (sometimes) drag & drop components to snapping locations
  - Java GUI builders; also Visual Studio docking (border layout)
- Struts & springs
  - Component edges are connected to container edges with either struts (fixed-size) or springs (variable-size)
  - Mac IB, Visual Studio
- Persistent alignment commands (Qt Designer)
  - Align horizontally, vertically, grid
  - Visible affordances after alignment (can be selected, edited to add padding, deleted)
  - "Spacer" components (visible during design, not at runtime)
- Snapping (Hudson & Yeatts, NetBeans 5.0)

Fall 2005 6.831 UI Design and Implementation 9

NetBeans 5.0's GUI Builder ("Matisse") uses a new layout manager, called GroupLayout, for expressing constraints among components. One thing GroupLayout does is vary its defaults according to the look-and-feel of the platform – so it can encode platform guidelines for sizes and spacing of widgets.

Widget Interface Conventions

- Java "Beans" convention was designed for GUI builders
  - getX(), setX()
  - addYListener(), removeYListener()
- GUI builder can inspect the methods of a class to create editable property list and event menu

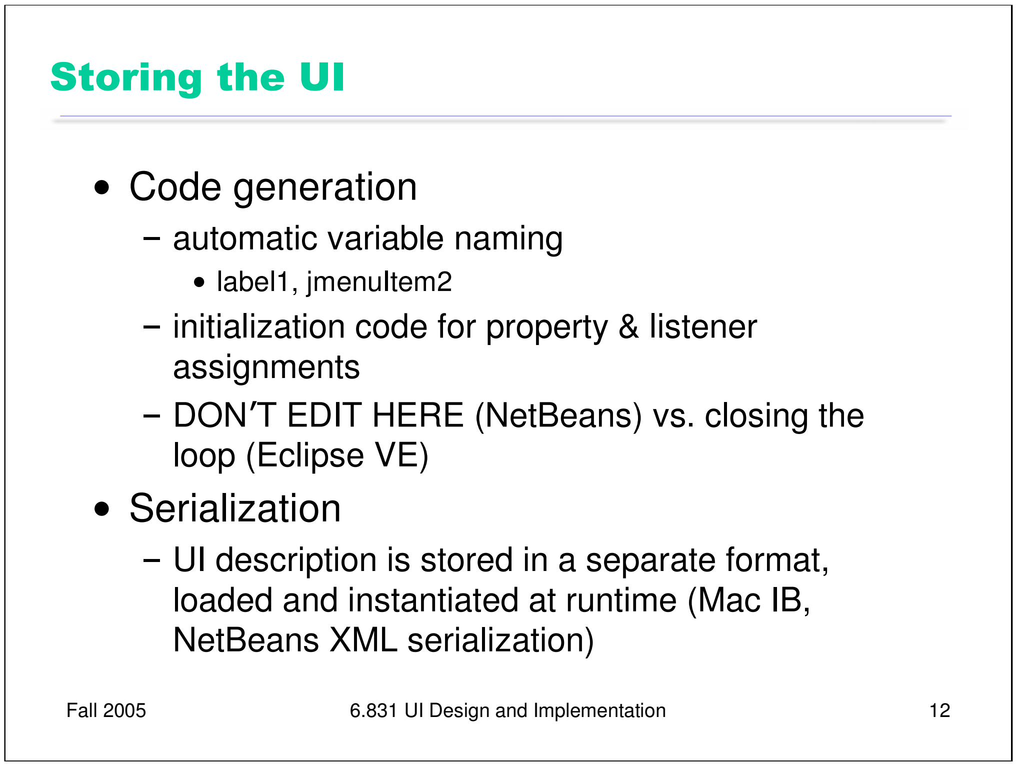


Modes

- Design mode
  - Clicking selects controls
  - Designer affordances are visible (e.g., selection handles)
- Test mode
  - Clicking operates the controls, but no backend (no event handlers)
  - Only user affordances are visible
- Run mode
  - Running the whole program, so backend works too

Fall 2005 6.831 UI Design and Implementation 11

Widgets



Storing the UI

- Code generation
  - automatic variable naming
    - label1, jmenuItem2
  - initialization code for property & listener assignments
  - DON'T EDIT HERE (NetBeans) vs. closing the loop (Eclipse VE)
- Serialization
  - UI description is stored in a separate format, loaded and instantiated at runtime (Mac IB, NetBeans XML serialization)