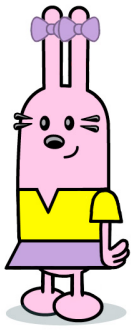
Widgets

Widgets

Widget Toolkits

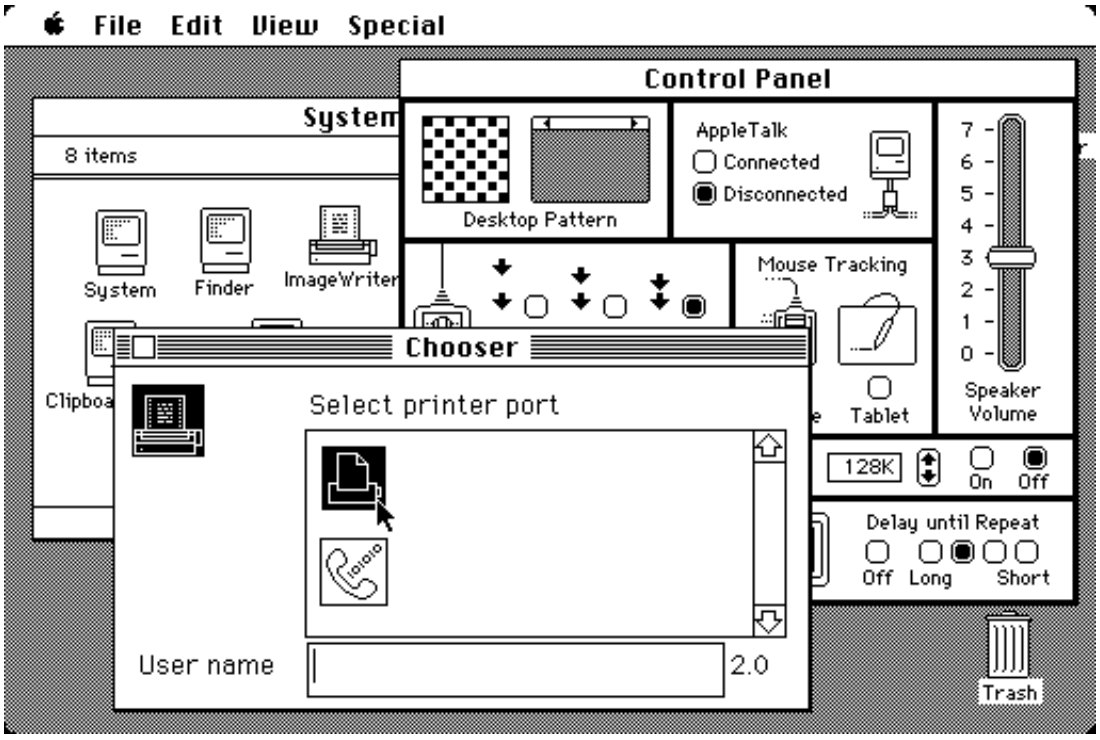
User Interface Widget

- **Widget** is a generic name for parts of an interface that have their own behavior: buttons, drop-down menus, spinners, file dialog boxes, progress bars, sliders, ...
- widgets also called **components**, or **controls**
- They provide user feedback and capture user input
- They have a defined appearance
- They send and receive events



Widget from ***Wow Wow Wubbzy***

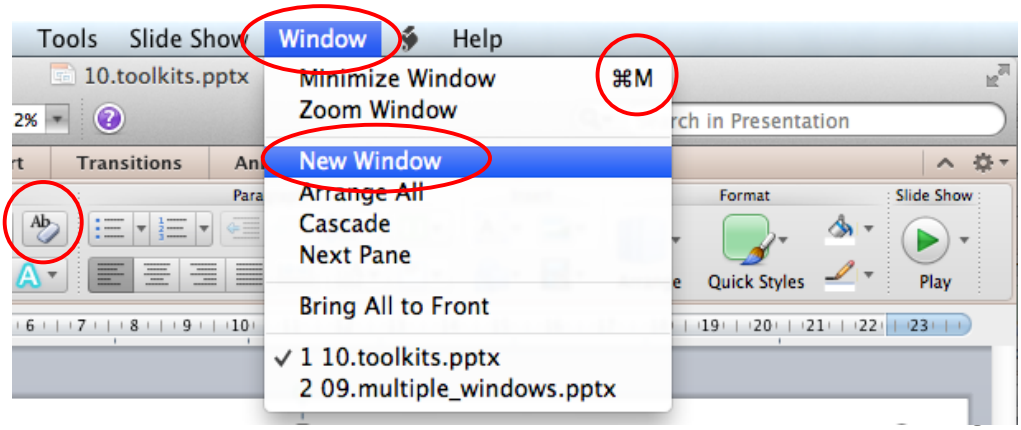
Early User Interface Widgets



Macintosh System 5, circa 1987

Logical Device

- A logical device is the essence of what a widget does, its *function*
- e.g. *logical button device*
  - function: generate “pushed” event
- A widget is a logical device with an appearance
- e.g. widgets based on *logical button device*
  - appearances: push button, keyboard shortcut, menu item, …

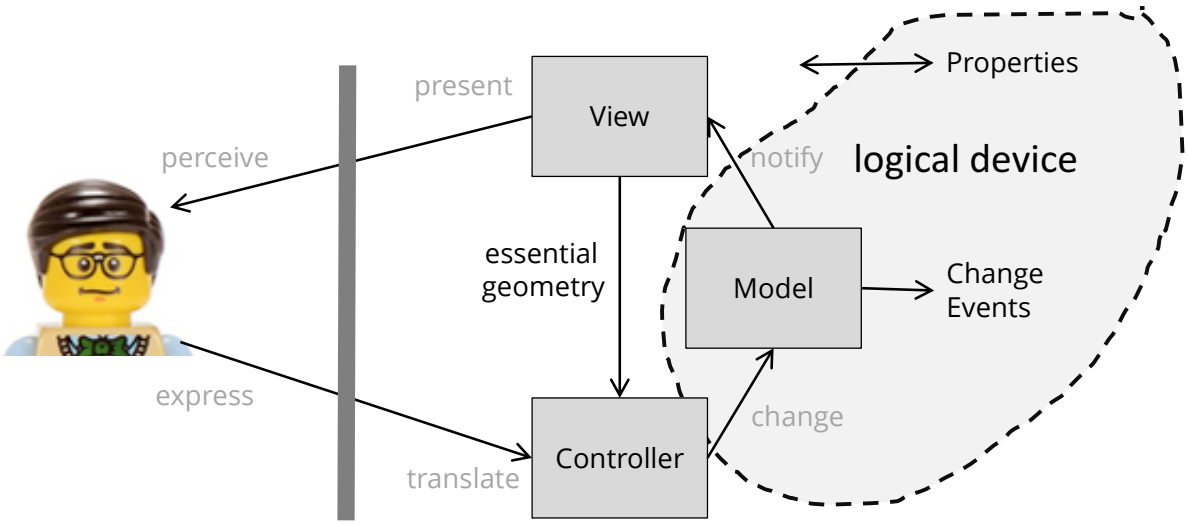


Other Logical Devices

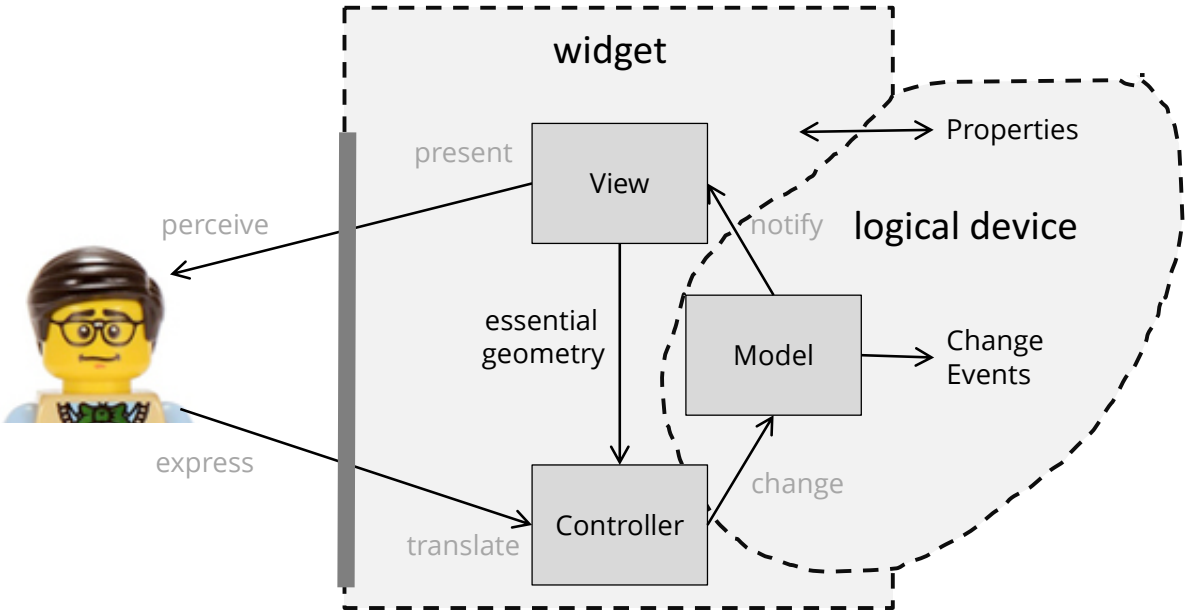
- *logical number device*
  - function: adjust a number, generates a “changed” event
  - appearances: slider, spinner, numeric textbox, …

- logical *boolean* device
  - function:
  - appearances:

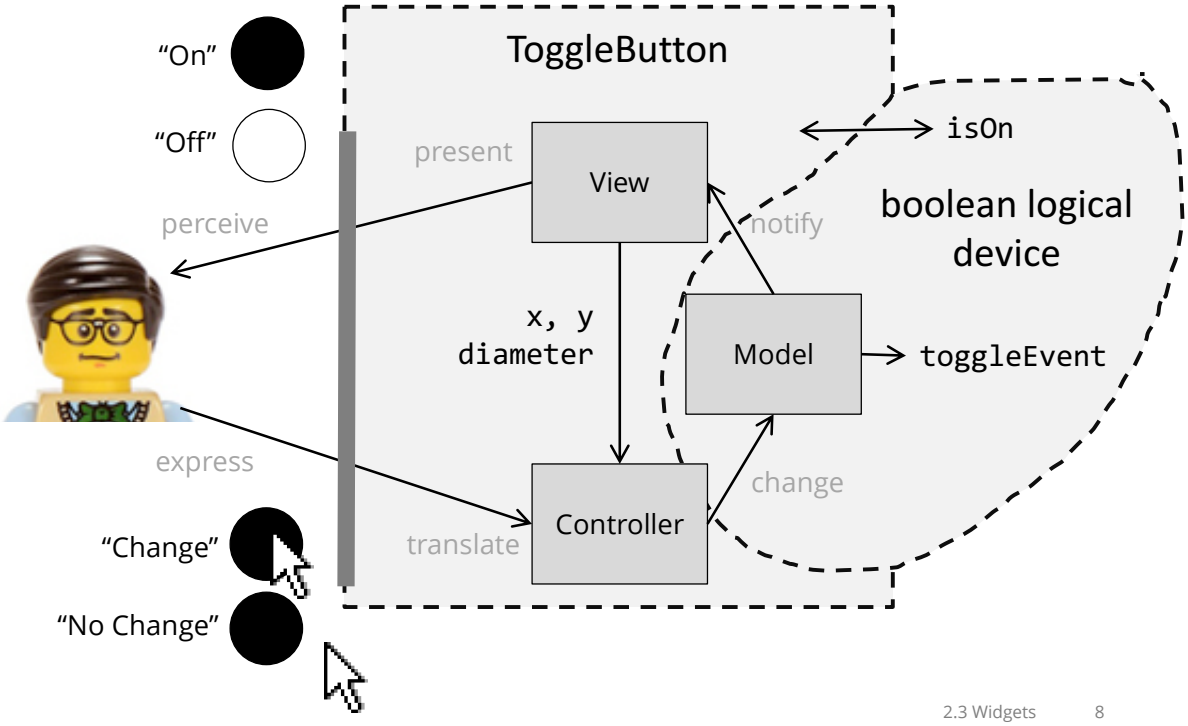
Widget Architecture as MVC



Widget Architecture as MVC



ToggleButton Widget Example



ToggleButton.cpp

```
class ToggleButton {
   ToggleButton(int _x, int _y, void (*_toggleEvent)(bool)) {
      toggleEvent = _toggleEvent;
      isOn = false;
      ...
   }

   // the CONTROLLER
   void mouseClick(int mx, int my) {
      float dist = sqrt(pow(mx - x, 2) + pow(my - y, 2));
      if (dist < diameter) { toggle(); }
   }

   ...
```


ToggleButton.cpp (cont'd)

```
   ...

   // the VIEW
   void draw() {
      if (isOn) {
         setForeground(BLACK);
         XFillArc(...);
      } else {
         setForeground(WHITE);
         XFillArc(...);
      }
   }

   // VIEW "essential geometry"
   int x;
   int y;
   int diameter;

   ...
```

ToggleButton.cpp (cont'd)

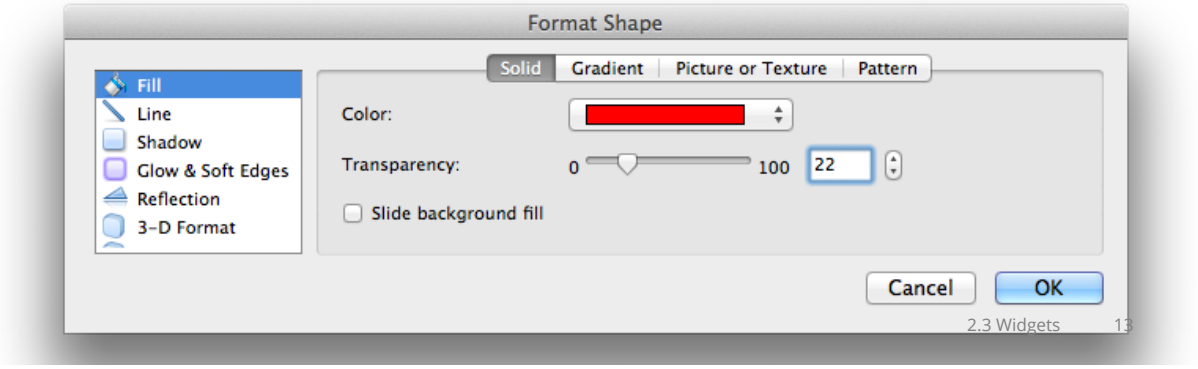
```
   ...

   // toggle event callback
   void (*toggleEvent)(bool);

   // the MODEL
   bool isOn;
   void toggle() {
      isOn = !isOn;
      toggleEvent(isOn); }
};
```


ToggleButton.cpp (cont'd)

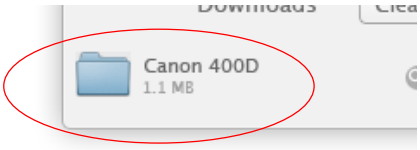
```
bool isPaused = false;
// isPaused callback (a simple event handler)
void togglePause(bool isOn) {
   isPaused = isOn;
}

...

ToggleButton toggleButton(150, 100, &togglePause);

...

   case ButtonPress:
      toggleButton.mouseClick(event.xbutton.x, event.xbutton.y);
      break;

...

if (!isPaused) {
   // update ball position
}

toggleButton.draw();
```

Categorizing and Characterizing Widgets

- Logical device
  (button, number, text, choice ...)
- Events the widget generates
  (action, change,...)
- Properties to change behaviour and appearance
  (colour, size, icon, allowable values, ...)
- Can it contain other widgets?
  (container vs. simple)



Simple Widgets



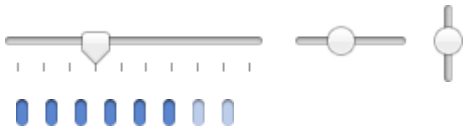
- Labels and Images
  - (usually) no model or events
  - e.g. label, icon, spacer,

- Button
  - no model, pushed event
  - properties: label, size
  - e.g. button

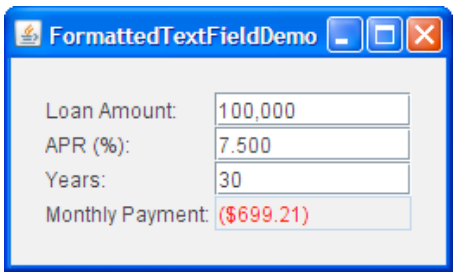
- Boolean
  - true/false model, changed event
  - e.g. radio button, checkbox, toggle button

“Radio Button”

Simple Widgets

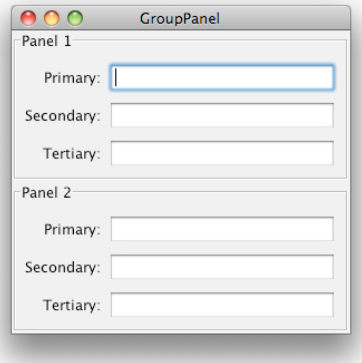


- Number
  - model: real number, changed event
  - properties: range, step
  - e.g. slider, progress bar, scrollbar

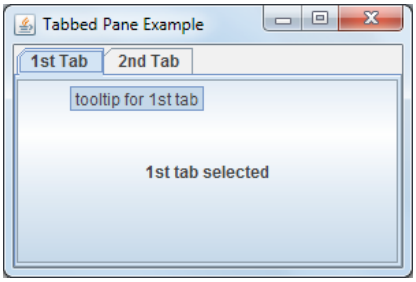


- Text
  - model: string; changed, selection, insertion events
  - properties: formatters (numeric, phone number, ...)

Container Widgets

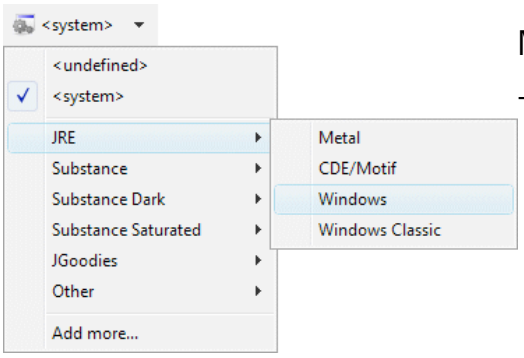


- Panel (Pane, Form, Toolbar)
  - arrangement of widgets
  - e.g. JPanel, toolbar

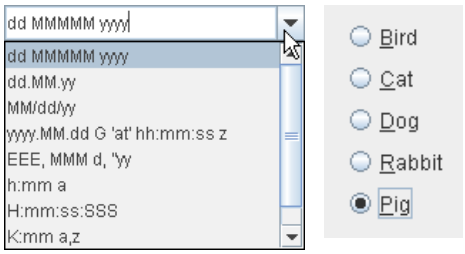


- Tab
  - choice between arrangements of widgets

Container Widgets



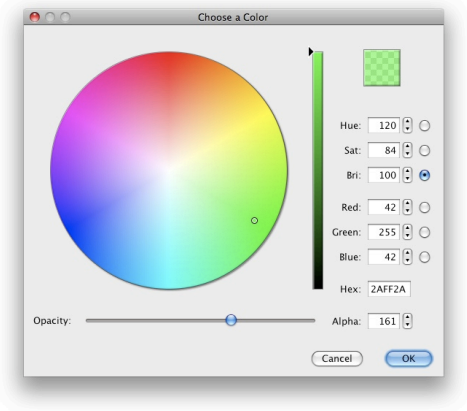
Menu
- hierarchical list of (usually) buttons

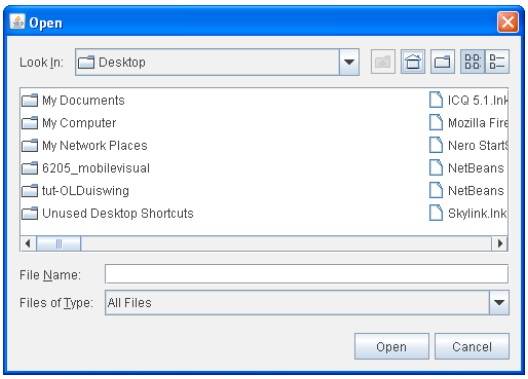


- Choice from a List
  - list of boolean widgets
  - e.g. drop-down, combo-box, radio button group, split button

Special Value Widgets

- colour/file/date/time pickers





Widget toolkits

- Also called widget libraries or GUI toolkits or GUI APIs
- Software bundled with a window manager, operating system, development language, hardware platform
- Defines a set of GUI components for programmers
  - Examples: buttons, drop-down menus, sliders, progress bars, lists, scrollbars, tab panes, file selection dialogs, etc.
- Programmers access these GUI components via an application programming interface (API)

Event-driven programming

- Widget toolkits use event-driven programming model
- Reactive systems
  - User action → program response
  - Most of the time the program sits around doing nothing
- Widget toolkit supports a mechanism for mapping user action on widget to appropriate application code to handle that action

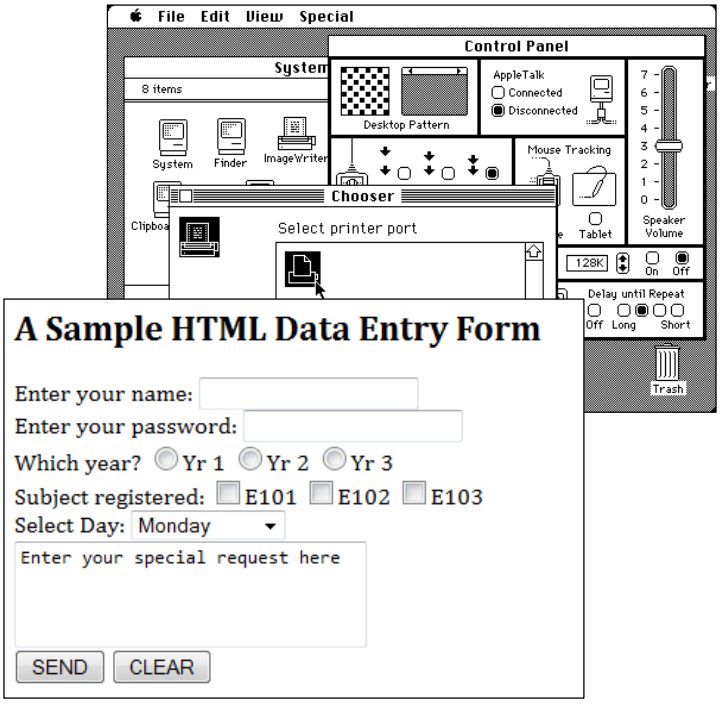
Widget Toolkit Design Goals:

- Complete
  - GUI designers have everything they need
- Consistent
  - Behaviour is consistent across components
- Customizable
  - Developer can reasonably extend functionality to meet particular needs of application

- Meeting these requirements encourages reuse

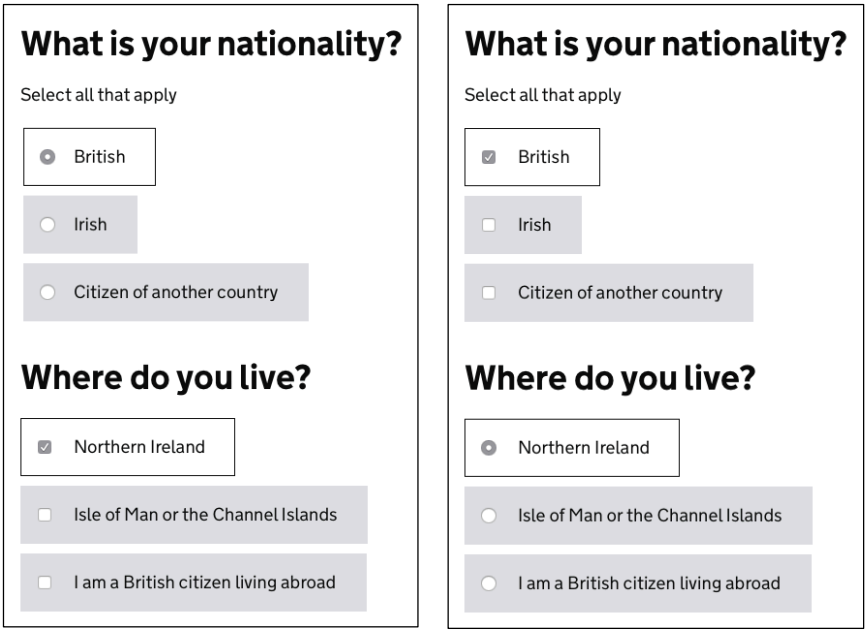
Completeness

- All you really need are:
  - Button
  - Slider
  - Pulldown menu
  - Check box
  - Radio button
  - Text field



Consistency

- Use a common look and feel
- Use widgets appropriately



Implementation Choices

- Heavyweight Widgets
  - OS provides widgets and hierarchical "windowing" system
  - Widget toolkit wraps OS widgets for programming language
  - BWS can dispatch events to a specific widget
  - *Examples: nested X Windows, Java's AWT, OSX Cocoa, standard HTML form widgets, Windows MFC*
- Lightweight Widgets
  - OS provides a top level window
  - Widget toolkit draws its own widgets and is responsible for mapping events to their corresponding widgets
  - *Examples: Java Swing, JQuery UI, WIndows WPF*

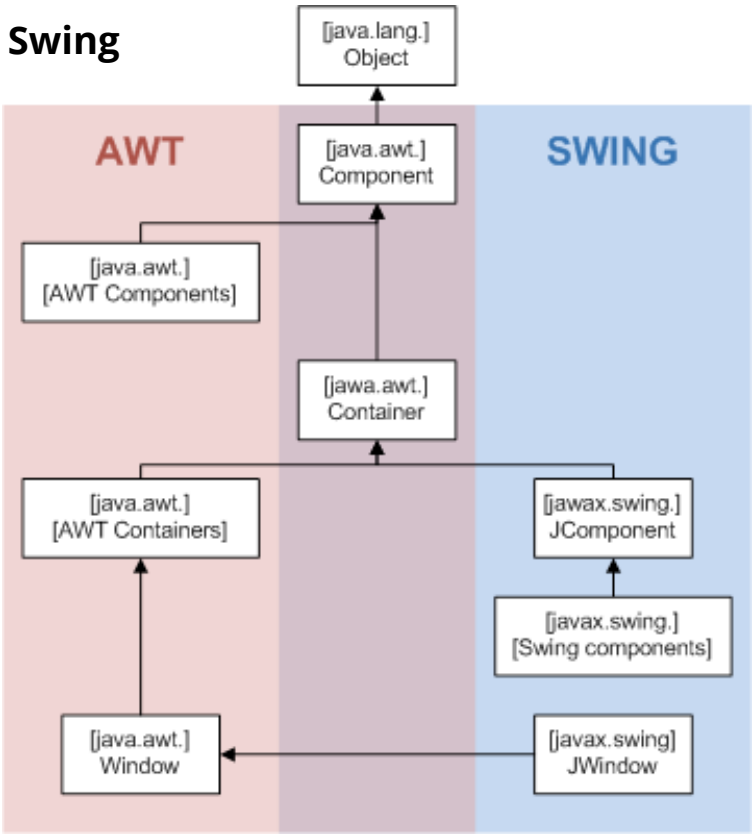
Java Abstract Window Toolkit (AWT)

- **Heavyweight toolkit**
  - OS standard widgets, mapped onto the Java language: Button, Canvas, Choice, Frame, Label, List, MenuBar, Panel, PopupMenu, Scrollbar, TextArea, Window
  - Since BWS is aware of them, can send events directly to them
- Only components that are supported on most platforms are included, so it's a minimal widget toolkit
  - the "least-common denominator" across OSX, Windows, Linux, …
  - so, no Spinner, no combo box, no progress bar, …
- Programmers need to re-create/find unsupported widgets
- Uses exact OS look and feel

Java Swing Widget Toolkit

- **Lightweight Toolkit**
- Widgets are implemented in Java 2D
  - essentially, custom draw commands in paintComponent()
- AWT is still required for Window (OS level)
- BWS only knows about main window, so widget event dispatch is done in Java
- Mixing Swing and AWT widgets *used to be* problematic
  - Fewer technical problems after Java 6
  - AWT rendered by OS, Swing rendered by Java (guess who won)
- Different Swing Look and Feels
  - (see SwingThemeDemo.java lecture code)

AWT and Swing



WidgetDemo.java

- Add JLabel, set properties
- Add JButton, JSlider
- Create JMenuItems, add to JMenu in a JMenuBar
- Create JRadioButtons, put into ButtonGroup and JPanel
- Create events for all widgets to setText in label
- Set layout manager for JFrame