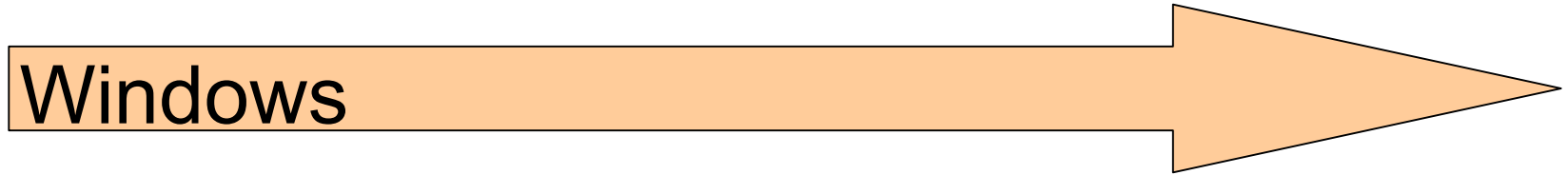


# WIMP Elements

GUI goo

# Outline

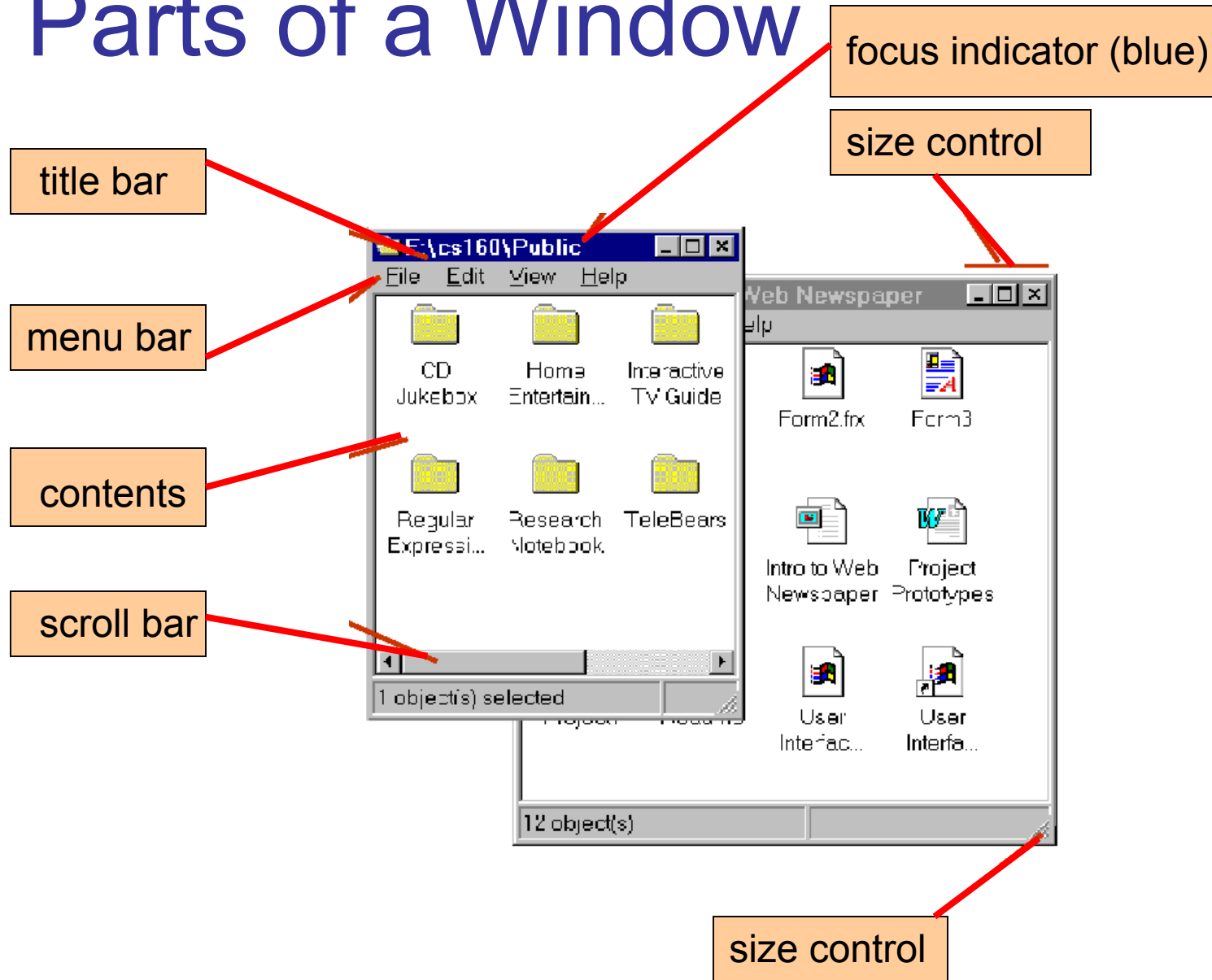
- Windows
- Icons
- Pointers
- Menus



# Windows

- Windows are areas of the screen that act like individual terminals for an application
- Behaviour of windows determined by the system's **window manager** (aka **windowing system**)
- Windows can contain text, graphics, menus, toolbars, etc.
- Can be moved, resized, closed, minimized, maximized

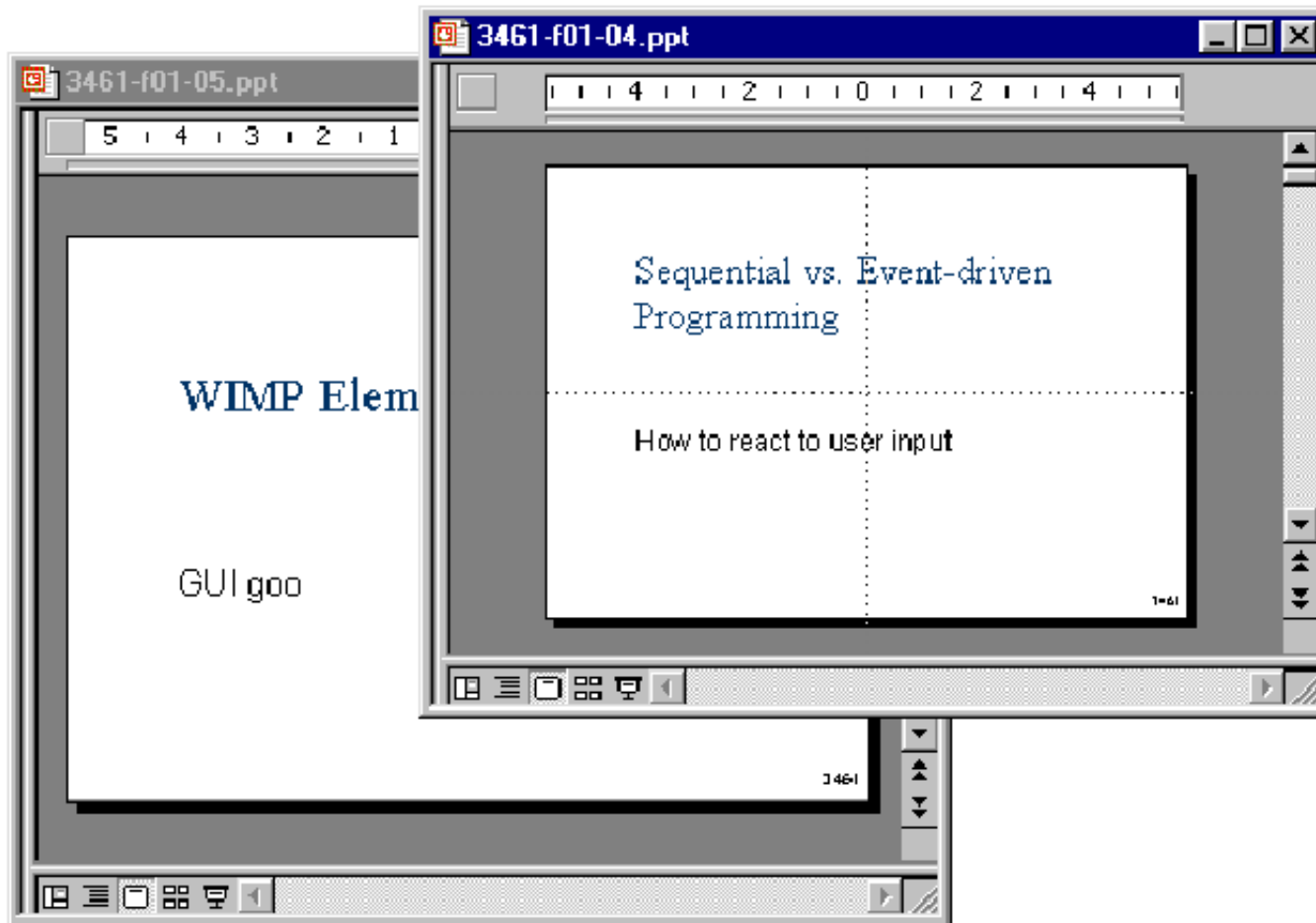
# Parts of a Window



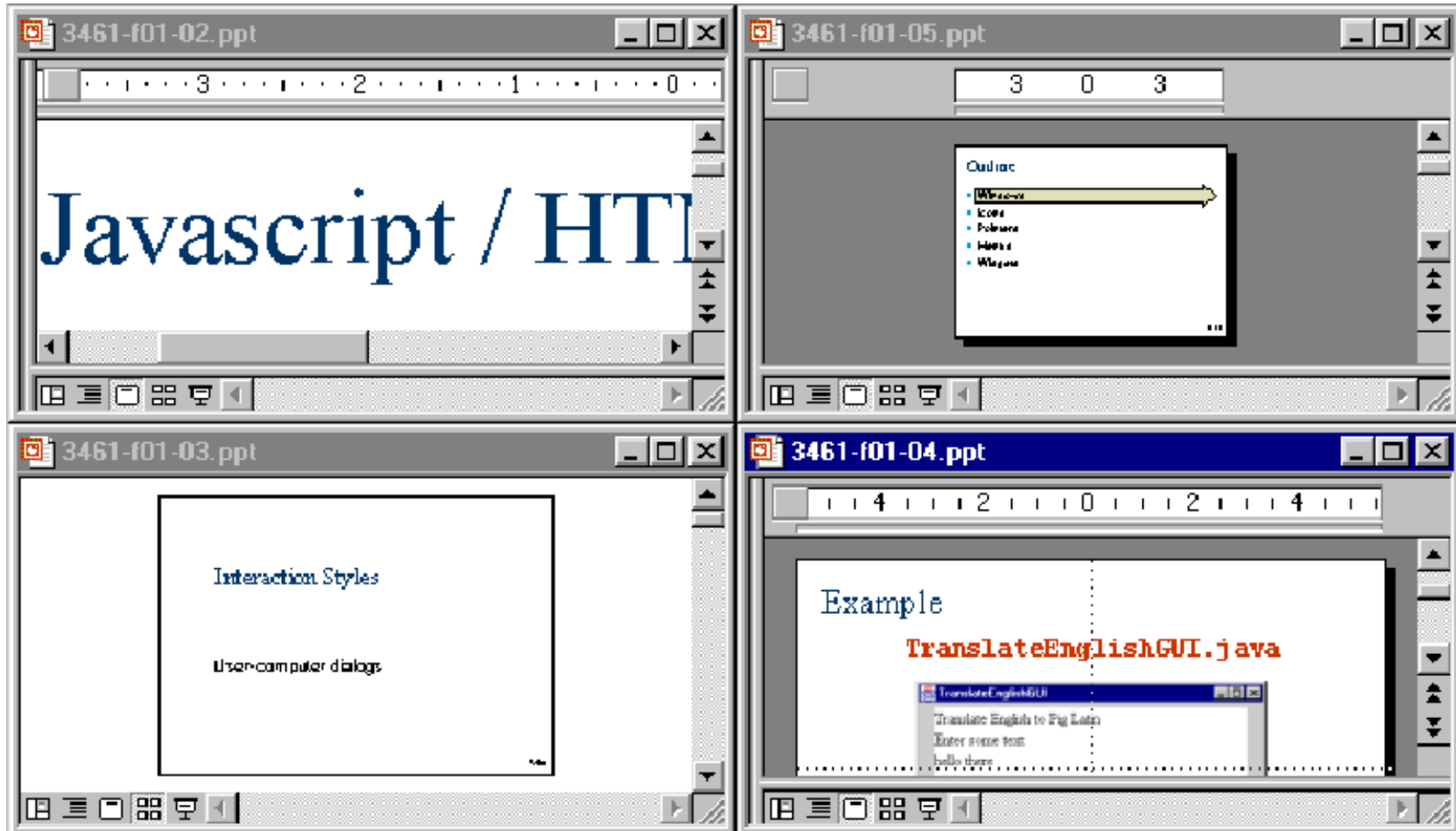
# Layout Policy

- Multiple windows may exist simultaneously
- Physical arrangement determined by the window manager's **layout policy**
- Layout policy may be fixed or user-selectable
- Possible layouts include
  - **Overlapping** - One window partially obscures another
  - **Tiled** - Adjoin but don't overlap
  - **Cascading** – A sequence with each window offset from the preceding according to a rule (e.g., 10 pixels to the right and below)
- Let's see...

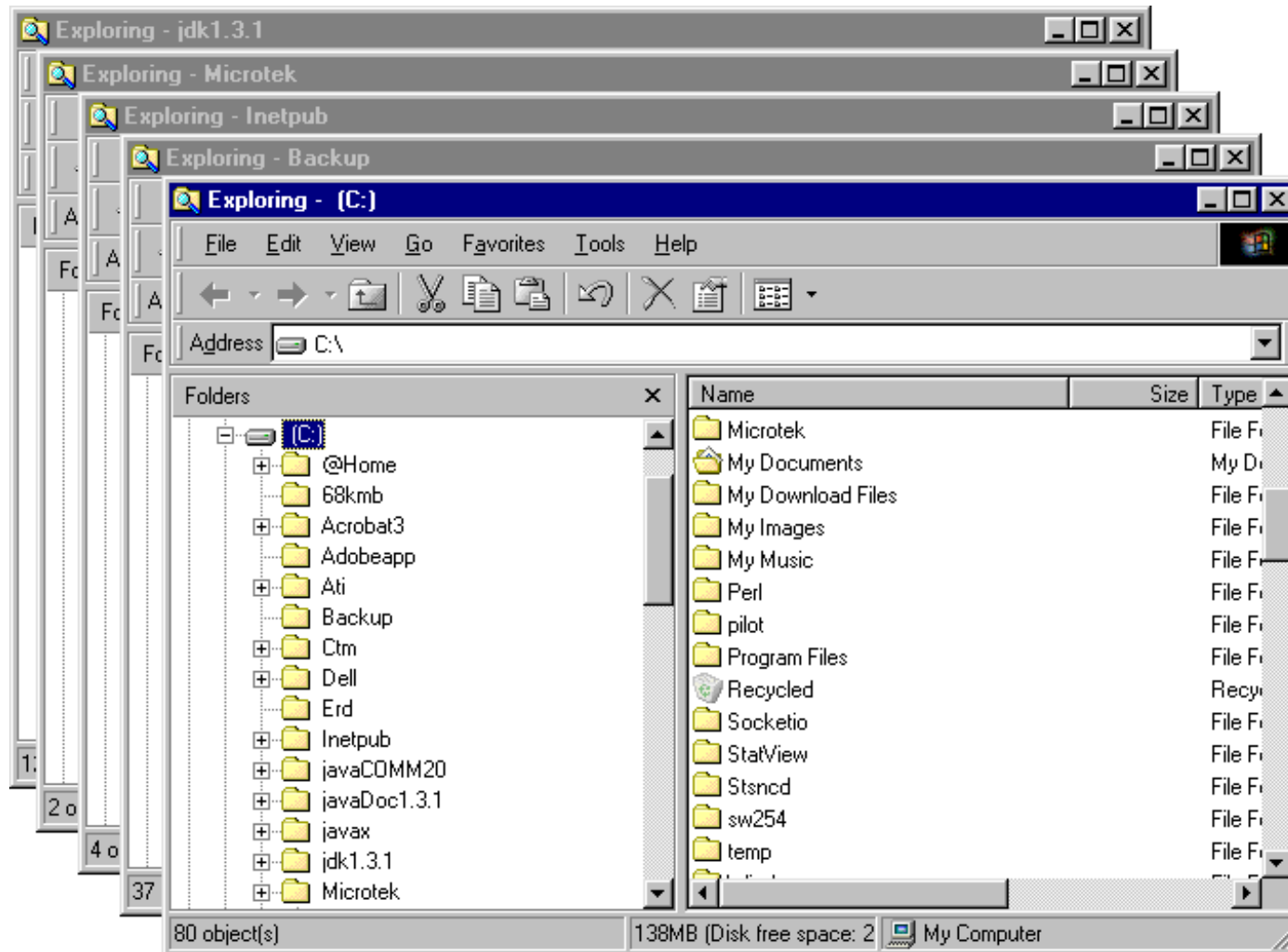
# Overlapping Windows



# Tiled Windows



# Cascading Windows



# Focus

- The active window is said to have **focus**
- Title bar of active window is blue (or some other distinct colour)
- Title bars of inactive windows are grey
- Clicking in an inactive window makes it the active window (i.e., gives it focus)
- Screen must be redrawn to bring the active window to the front

# Window Size States

- Windows have three size states
  - **Maximized**
    - Fills available space
  - **Minimized**
    - Reduced to a title bar or icon
  - **Normal (aka Restored)**
    - This is the size of the window, either when it was first opened, or before the window was maximized
    - From this mode, the window width and height may be adjusted

# Window Size Control (*Windows*)

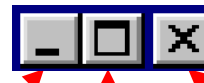
- Via boxes in upper-right corner of window

- When **maximized**...



Minimize Restore Close

- When **restored**...



Minimize Restore Close

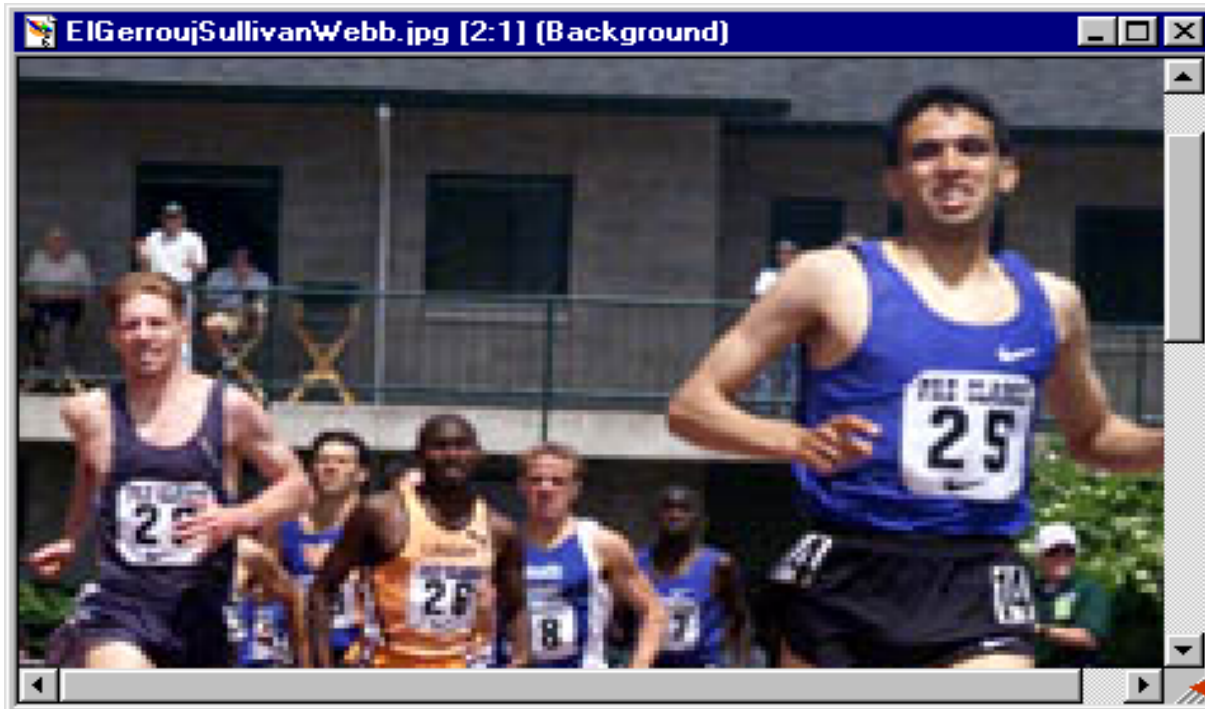
- When **minimized**...



Minimize Restore Close

# Window Size Control (2)

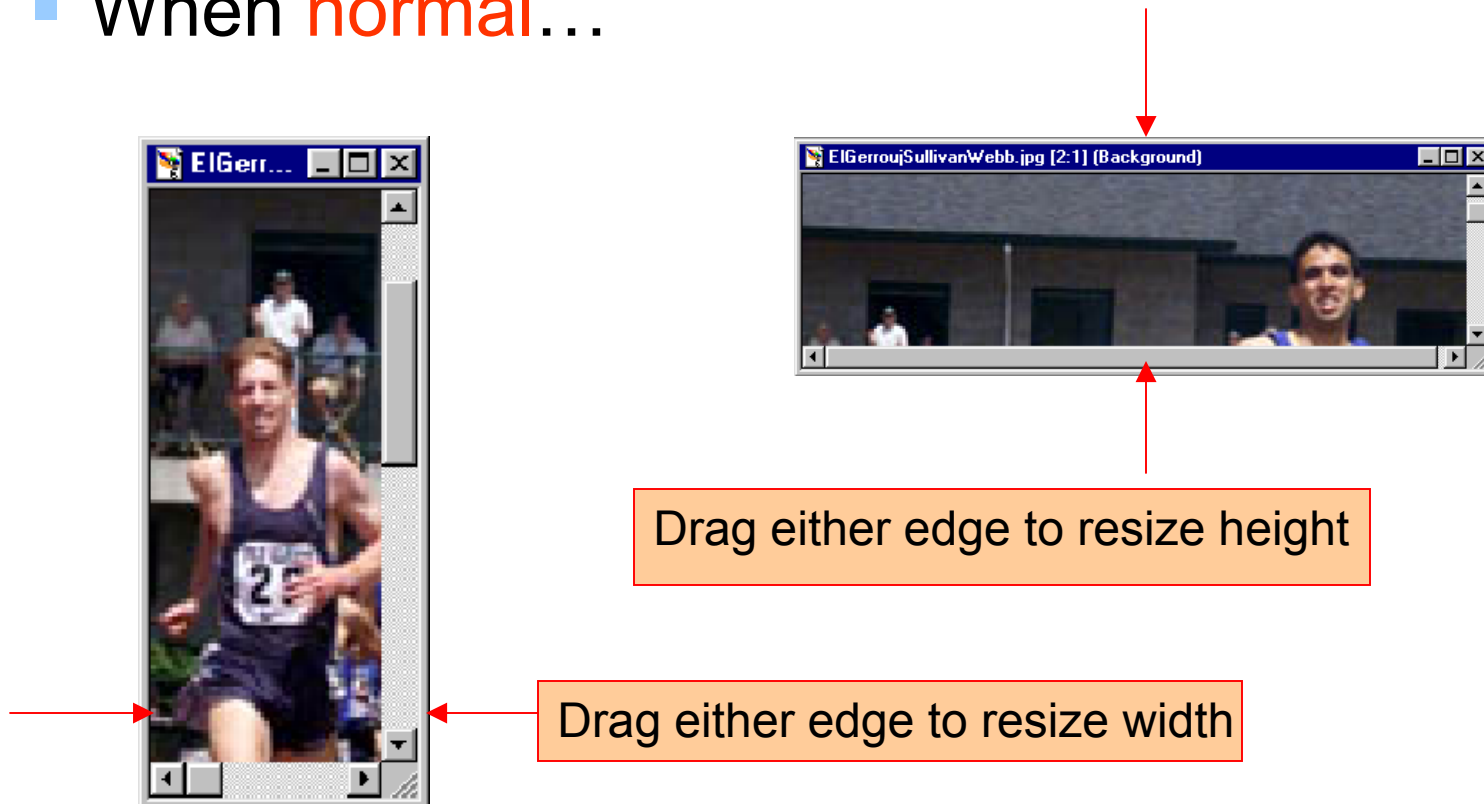
- Via handle in lower-right corner of window
  - When **normal**...



Drag to resize

# Window Size Control (3)

- Via virtual handles on edges
  - When **normal**...



# Window Managers

- User interfaces are typically implemented within the framework of a **window manager**
  - Also known as **windowing system** or **user interface management system (UIMS)**
- Provides
  - Partitioning to prevent chaos on the screen (What if there was only one window shared by all applications?)
  - Layout policy
  - Infrastructure to support common services in building UIs

# Window Manager Structure

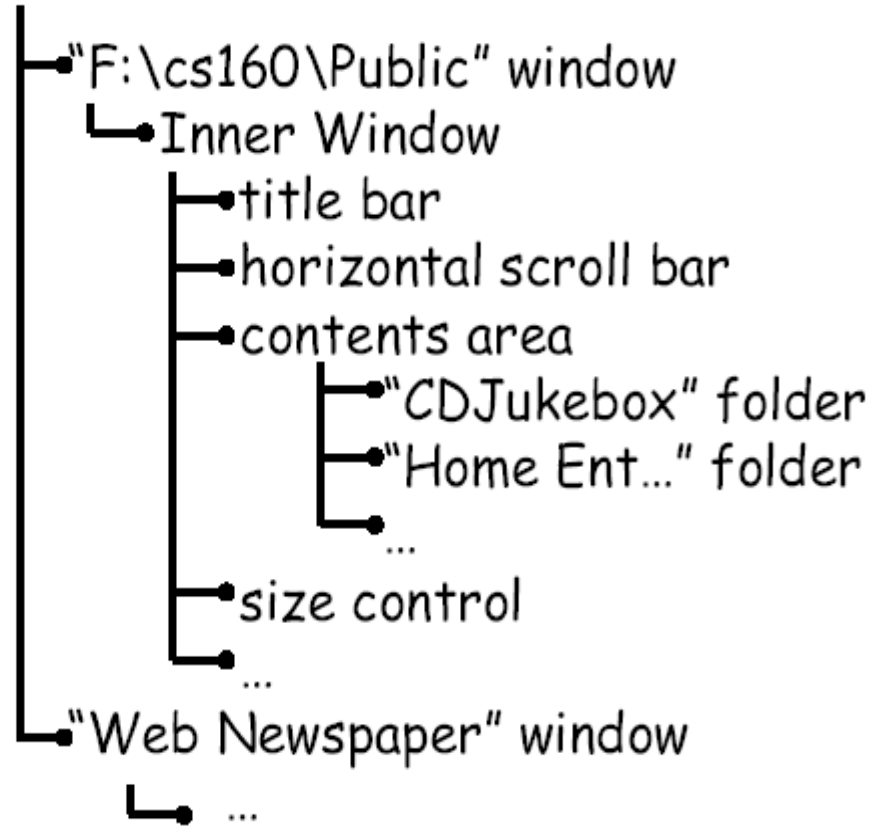
- Base layer (implements the basic infrastructure)
  - Output model (graphics primitives)
  - Input model (keyboard, mouse)
- UI layer (handles all visible aspects)
  - Presentation (e.g., what is on top?)
  - Commands (window & content manipulation)
- Mapping of input actions to applications
- When building a UI, use services of windowing system where possible (rather than writing custom code)

# Containment Hierarchy

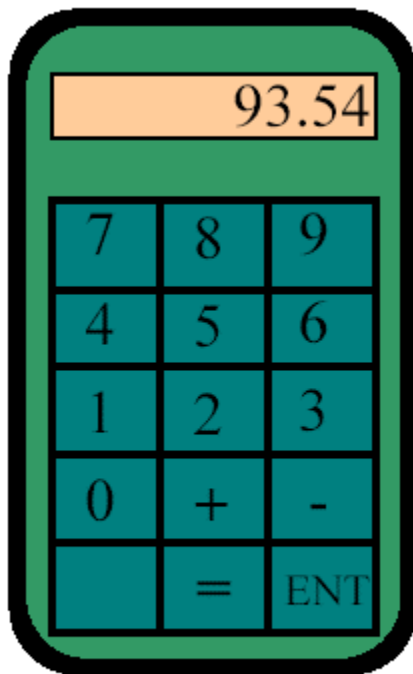
- A window contains a number of nested interactive objects (e.g., buttons, text fields, other windows)
- Relationship of objects is expressed by a **containment hierarchy** (aka **interactor tree**)
  - based on screen geometry of objects
  - represents the hierarchy/nesting of the objects

# Containment Hierarchy - Example 1

Display Screen



# Containment Hierarchy - Example 2



Display Screen

- ↳ Outer Win [*black*]

- ↳ Inner Win [*green*]

- ↳ Result Win [*tan*]

- ↳ Result String

- ↳ Keypad [*Teal*]

- ↳ = button

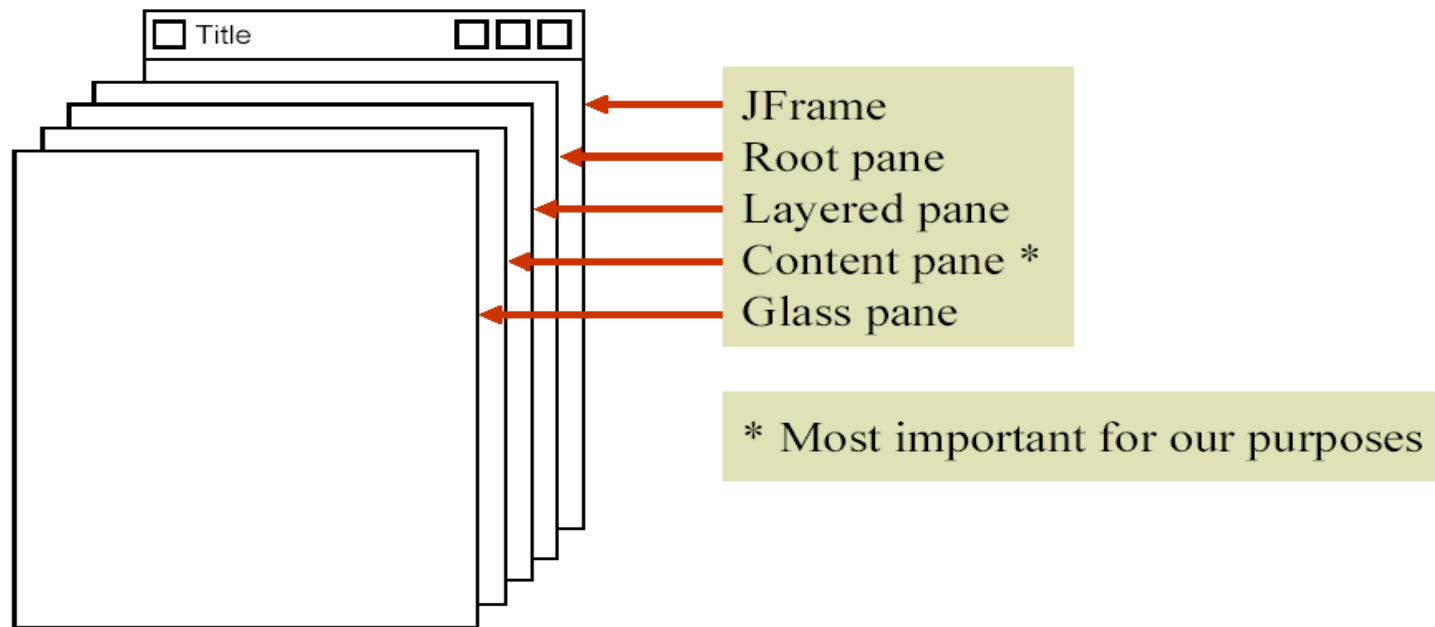
- ↳ - button

- ↳ + button

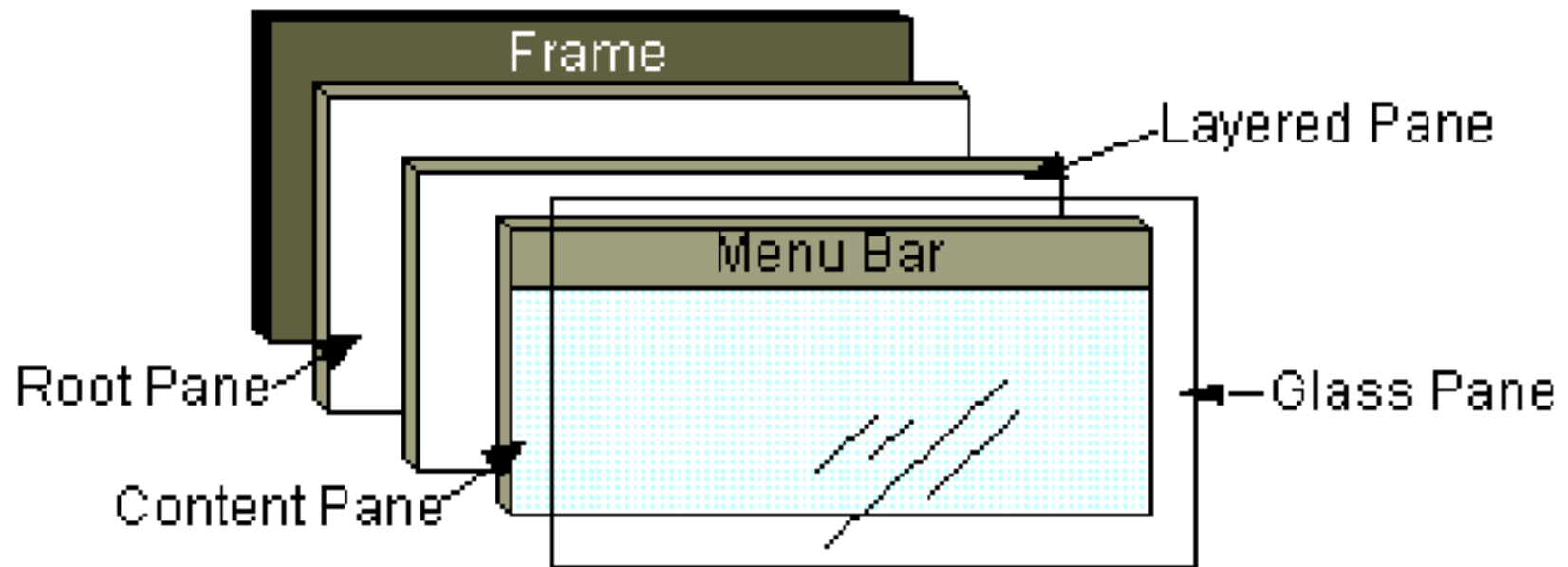
- ↳ 0 button

# Java's Containment Hierarchy

- With JFC/Swing, the basic building block for a window is the JFrame and its associated panes



# Java's Containment Hierarchy (2)



See "Using Top-Level Containers" in the Swing Tutorial

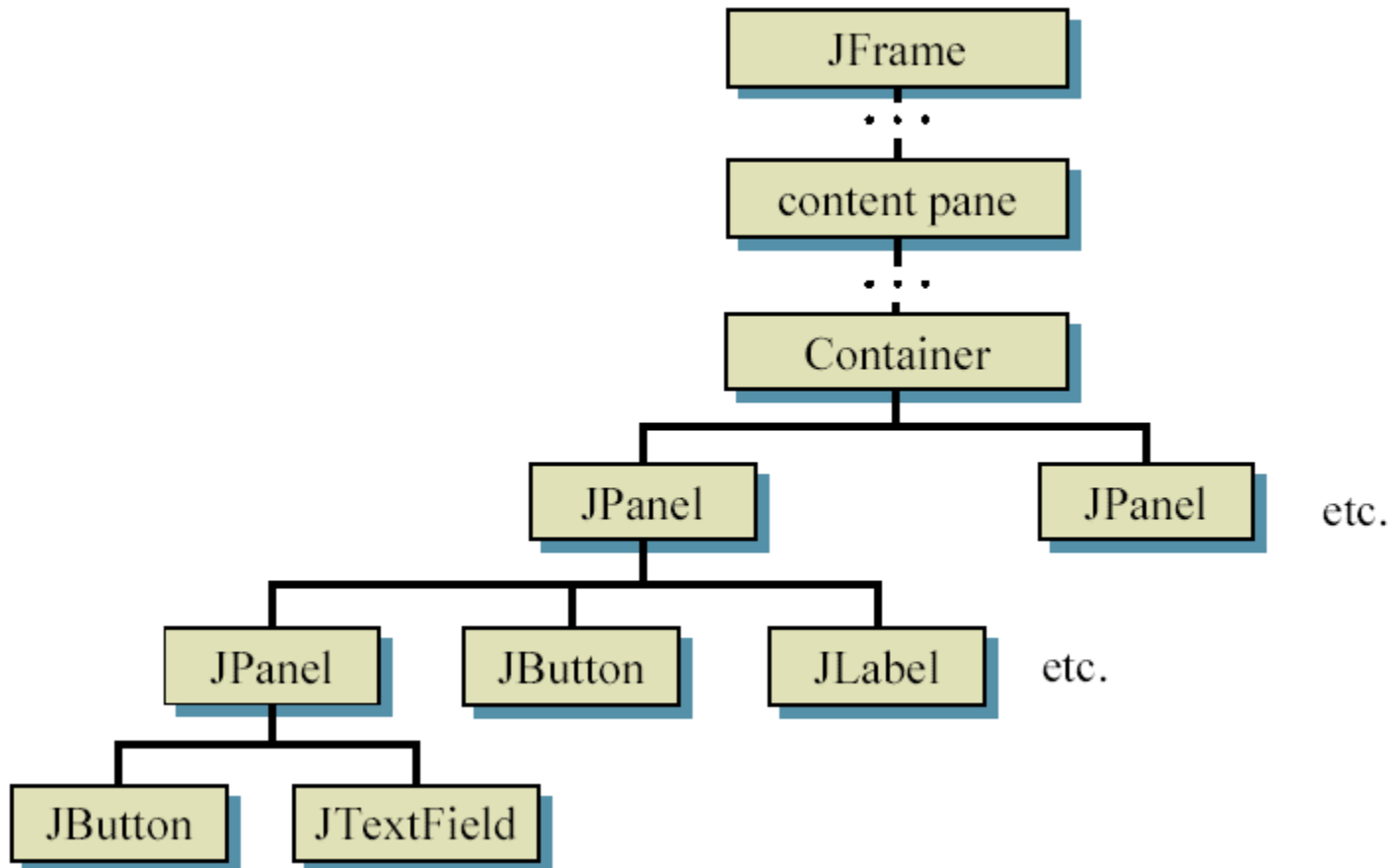
# Containers

- Components are placed in **containers**
- A JFrame is a **top-level container**
  - It exists mainly as a place for other components to paint themselves
  - Other top-level containers are dialogs (JDialog) and applets (JApplet)
  - Cannot place a JFrame inside a JFrame
- A JPanel is an **intermediate container**
  - Sole purpose is to simplify the positioning of interactive objects, such as buttons or text fields
  - Other intermediate containers are scroll panes (JScrollPane) and tabbed panes (JTabbedPane)
  - Can place a JPanel inside a JPanel (or inside a JFrame, via the content pane)

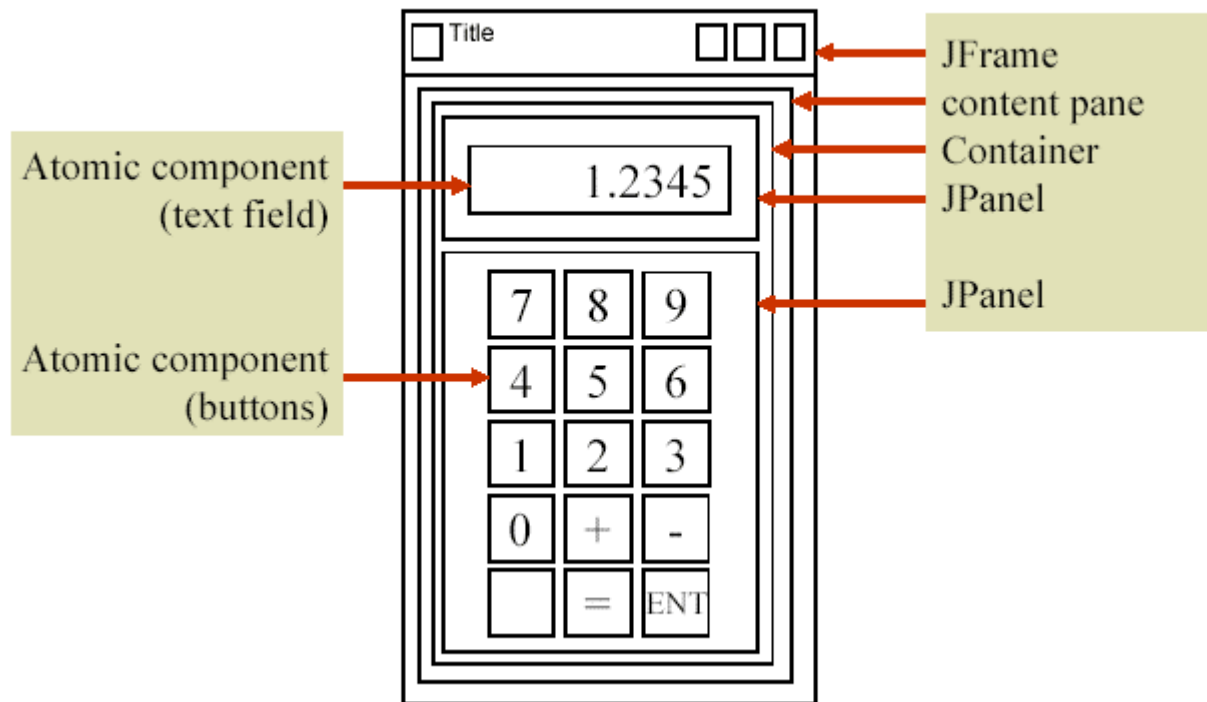
# Atomic Components

- An **atomic component** is a component that exists solely to present and perhaps accept information
- Examples: buttons (JButton), text fields (JTextField), combo boxes (JComboBox)
- JFrame and JPanel are also components, however...
  - They can hold other components
  - An atomic component cannot hold other components

# Containment Hierarchy for JFC/Swing

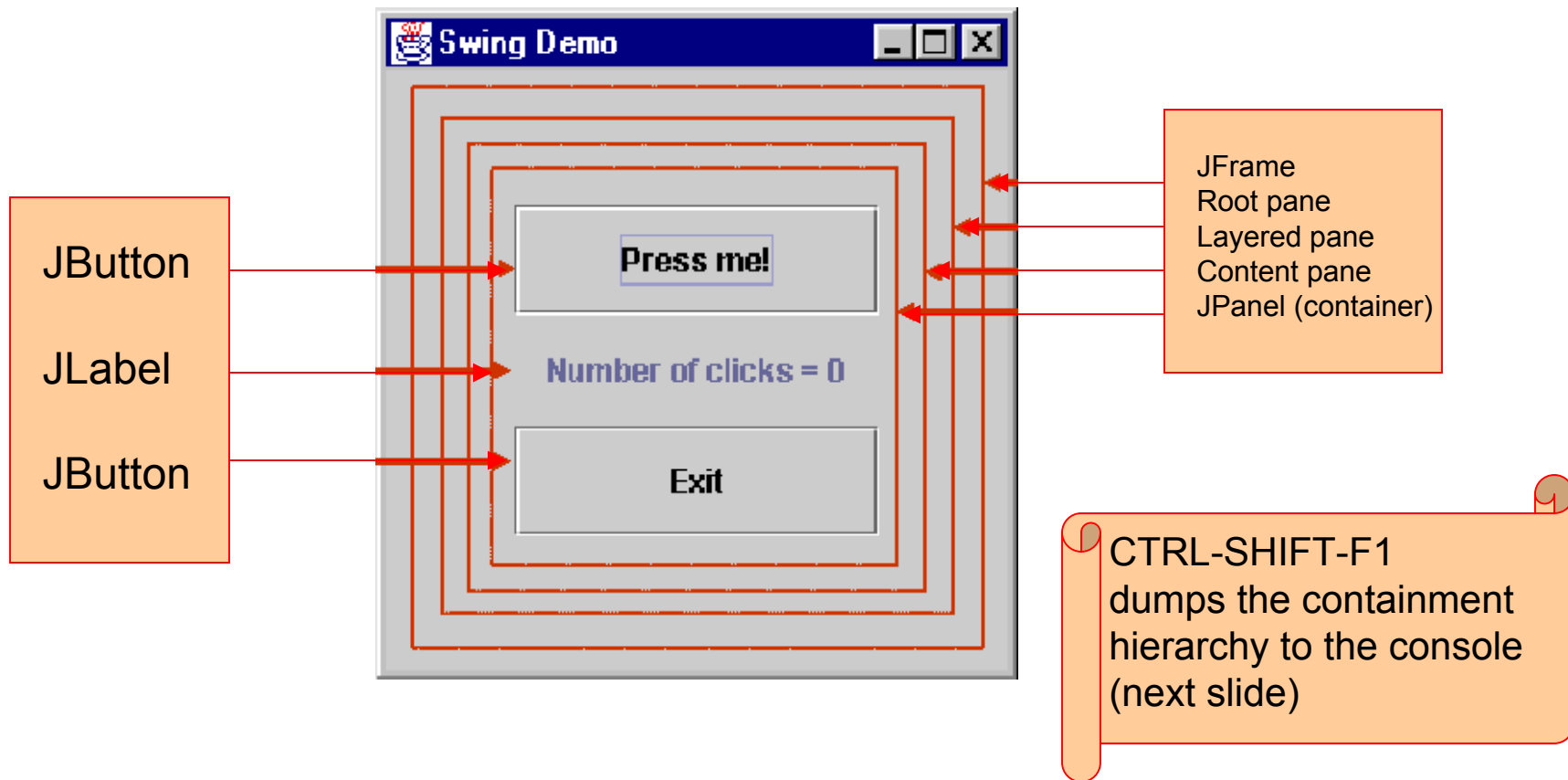


# So...



# Example Program

## DemoSwing.java



# Containment Hierarchy (abbreviated) for DemoSwing.java

```
DemoSwingFrame[frame0,0,0,121x128, ...  
  javax.swing.JRootPane[,4,23,113x101, ...  
    javax.swing.JPanel[null.glassPane,0,0,113x101, ...  
      javax.swing.JLayeredPane[null.layeredPane,0,0,113x101, ...  
        javax.swing.JPanel[,0,0,113x101, ...  
          javax.swing.JButton[,10,10,93x27, ...  
            javax.swing.JLabel[,10,37,93x27, ...  
              javax.swing.JButton[,10,64,93x27, ...
```

# Outline

- Windows
- Icons
- Pointers
- Menus
- Widgets



# What is an Icon

- From Webster's dictionary:
  - Icon: *a pictorial representation*
- A window may be closed and lost forever, or...
  - Shrunk to a reduced representation
  - The reduced representation is called an icon
- The act of reducing a window to an icon is called **iconifying** or **minimizing**
- A window may be restored by clicking on its icon
- Advantages of icons...
  - Save screen space
  - Serve as a reminder of available dialogs, applications, or commands that may be restored or invoked

# Icons Are Used to Represent...

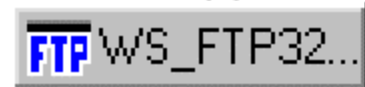
- Disk drives



- Available applications



- Mimized applications



- Minimized windows



- Folders



- Files



- Commands

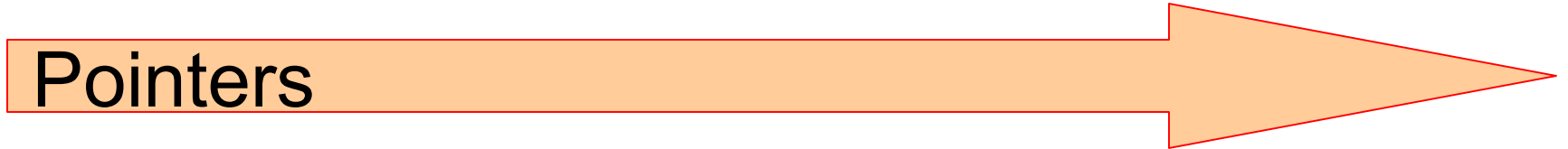


- States



# Outline

- Windows
- Icons
- Pointers
- Menus



# What is a Pointer?

- A pointer is the input device used to interact with GUI components
  - E.g., mouse, trackball, joystick, touchpad, finger, stylus, light pen
- Two primary purposes
  - **Position control** of the on-screen tracker
  - **Selection** via buttons

# Direct vs. Indirect Input

- Direct input
  - Via finger, stylus, light pen
  - No spatial displacement between input device and display
  - Tracker generally not needed
  - Selection via tapping or pressing
- Indirect input
  - Via mouse, joystick, or trackball
  - Spatial displacement between input device and display
  - Tracker needed
  - Selection via button presses

# Selection Primitives

- Generally, at least two buttons on pointing devices
- Selection primitives
  - Primary button (default = left)
    - Single click – select
    - Double click – launch
    - Drag – select region
  - Secondary button (default = right)
    - Click – invoke context-sensitive menu

# Tracker

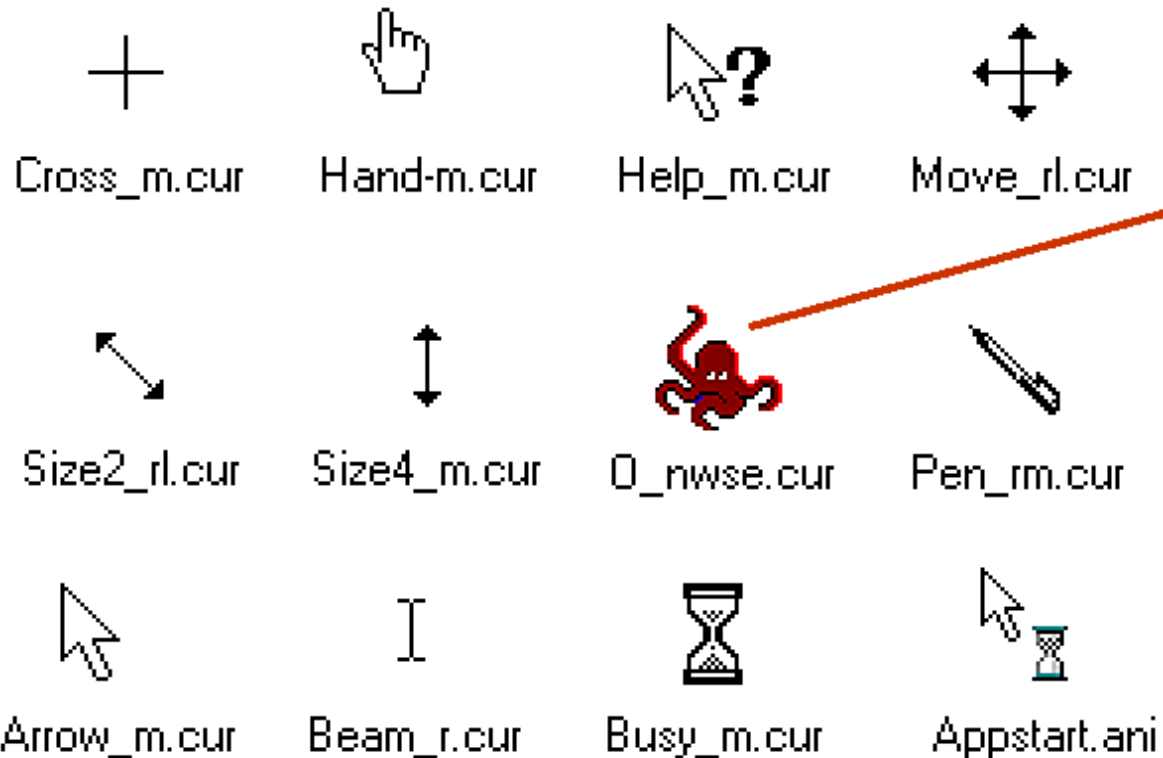
- The on-screen symbol that follows (“tracks”) the motion of the input device is called a **tracker** (aka **cursor**)
- Two primary purposes
  - **Position indicator** – crucial feedback for input control
  - **State indicator** – reveals current state of the system or GUI component

# Tracker Hot Spot

- The tracker is a bit-mapped image ( $x$  by  $y$  pixels)
- One pixel in the image is defined as the hot spot
- Selection occurs at the coordinate of the hot spot
- When designing custom trackers, use an image with an obvious hot spot if selection is required while the tracker is displayed

# Tracker Examples

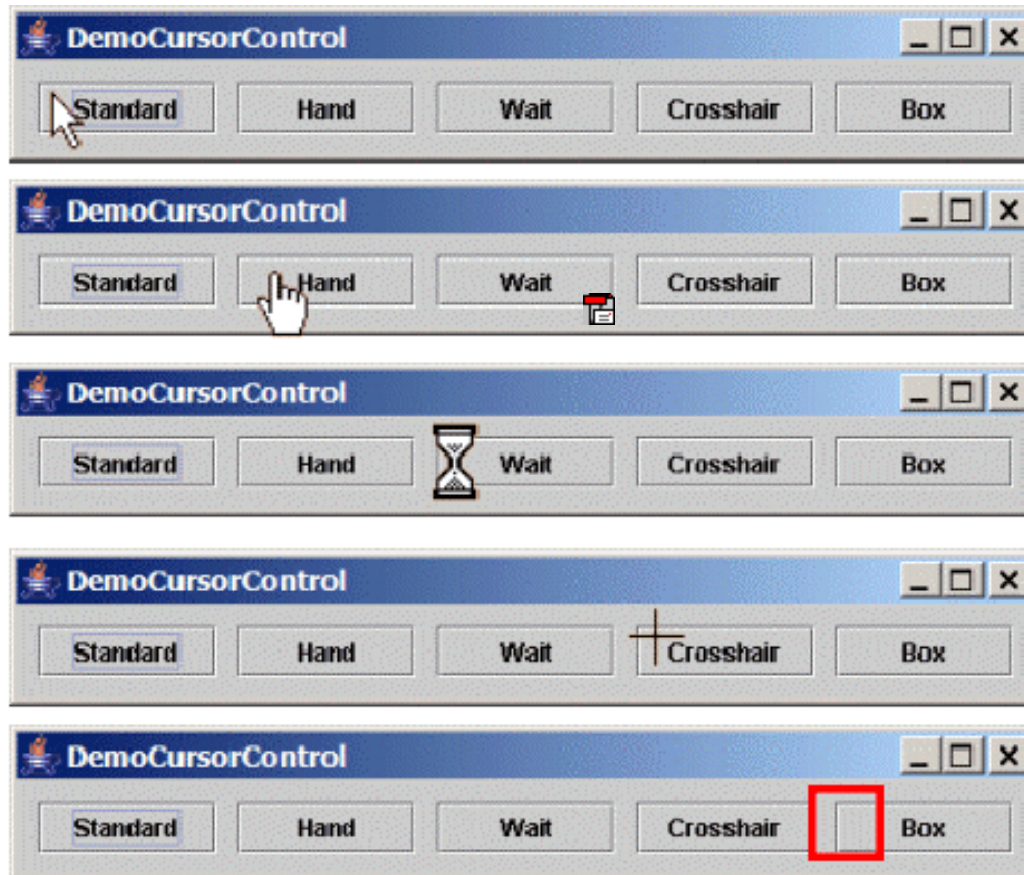
- Examples from MS Windows



Where is the hot spot?

# Example Program

## DemoCursorControl.java





# Outline

- Windows
- Icons
- Pointers
- **Menus**



# What is a Menu?

- A means of presenting a choice of operations that can be performed by the system at a given time
- Main advantage:
  - Menu options are **recognized** rather than **recalled**
  - Human ability to recognize is superior to ability to recall
  - Example of recall: *Who is the captain of the Maple Leafs?*
  - 
  - 
- Menus typically navigated two ways
  - Keyboard
  - Pointing device

# What is a Menu?

- A means of presenting a choice of operations that can be performed by the system at a given time
- Main advantage:
  - Menu options are **recognized** rather than **recalled**
  - Human ability to recognize is superior to ability to recall
  - Example of recall: *Who is the captain of the Maple Leafs?*
  - Example of recognition: *The captain of the Maple Leafs is (a) Tie Domi (b) Matts Sundin, (c) Darcy Tucker, (d) Steven Harper.*
- Menus typically navigated two ways
  - Keyboard
  - Pointing device

# Menu Location

- Most application windows include a menu bar directly below the title bar

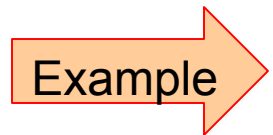


# Menu Design Techniques

- Numerous techniques are used to design effective menus
- Many are accompanied by **visual indicators**
  - Serve as signal to the user
- Menu features
  - Cascading submenus
  - Groupings
  - Dialog boxes
  - Icons
  - Keyboard input
    - Mnemonics
    - Accelerators
  - Popup menus

# Cascading Menus

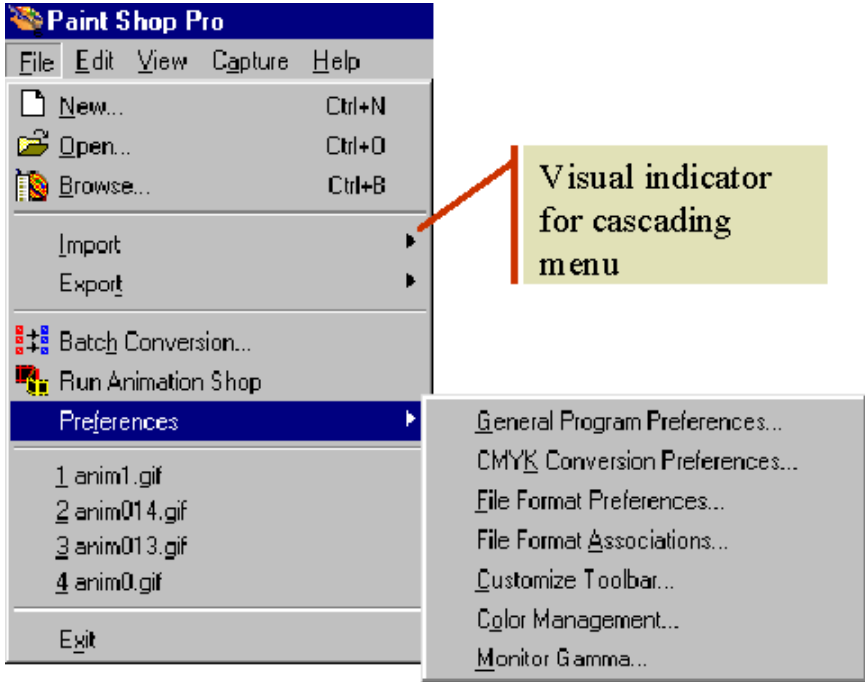
- Menus are inefficient if they contain too many items
- One solution is to use **cascading menus** (aka **submenus**)
- Selecting an item opens up another menu adjacent to selected item
- Several layers of cascading menus may be used
- Visual indicator: triangle
- Example



## File menu

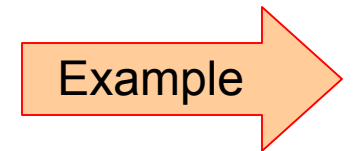


## File menu with focus on Preferences



# Groupings

- Similar items are grouped together in a menu
- Visual indicator: separator (i.e., line)



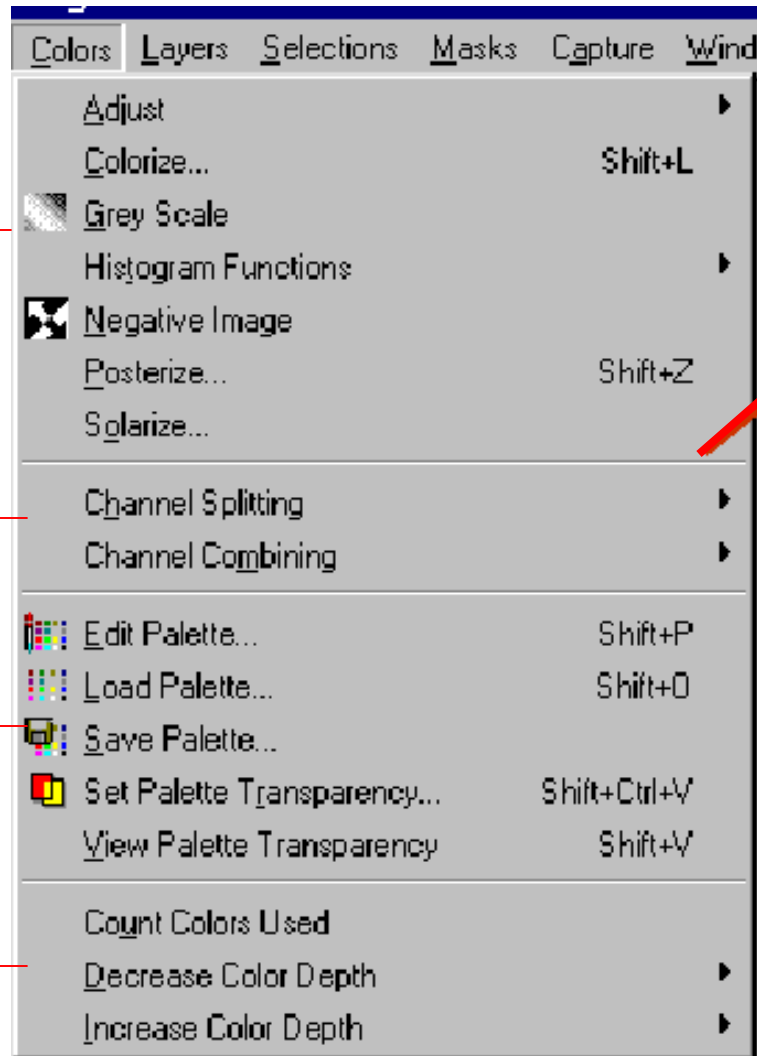
## Color menu

Image adjust group

Channel group

Palette group

Color depth group



Visual indicator for group

# Dialog Boxes

- A Menu choice that involves the collection of input parameters can use a dialog box
- May contain a message, editable fields, buttons, etc.
- Types of dialog boxes
  - File (open, new, save as)
  - Print
  - Color chooser
- Visual indicator: Ellipsis (...)



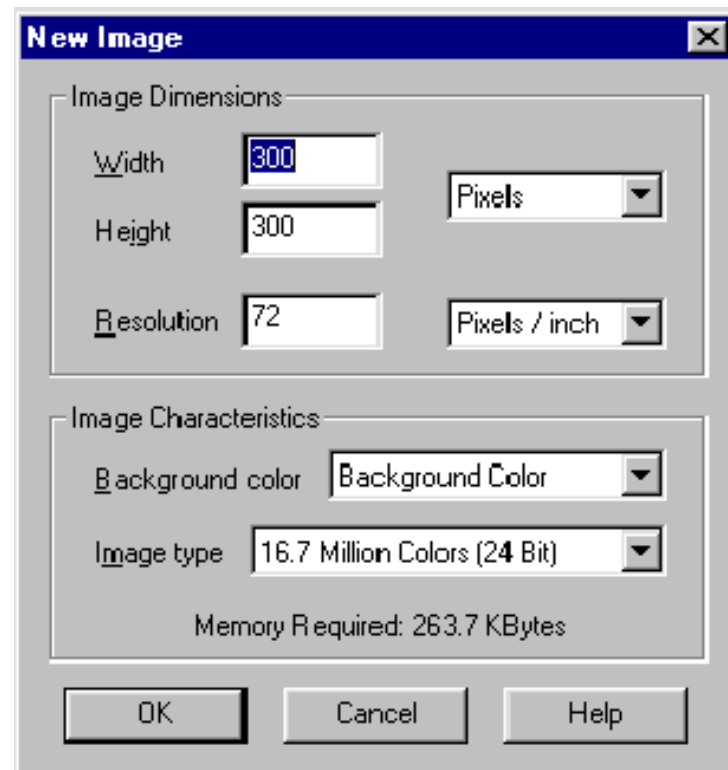
Example

File menu


Visual indicator  
for dialog box



“New” dialog box



# Icons in Menus

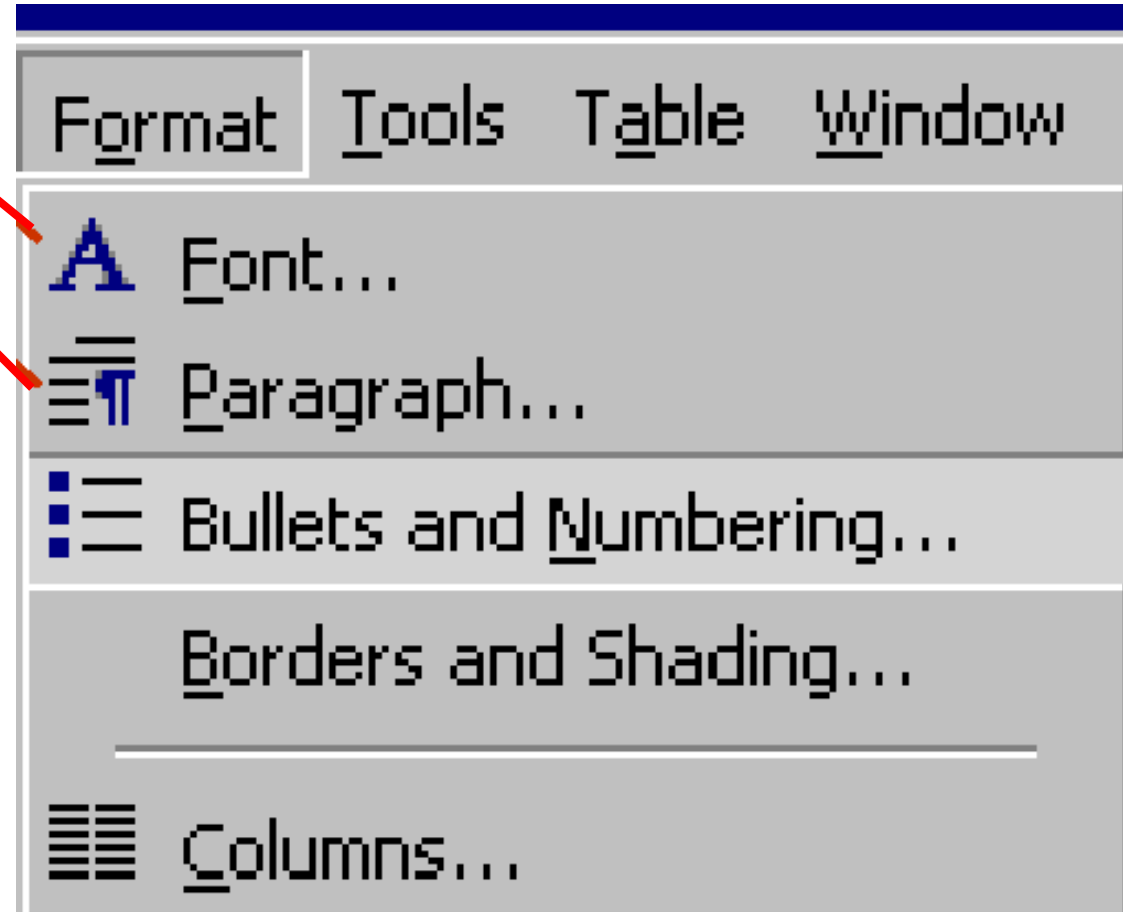
- Menu items typically contain words as labels
- Two problems with words as labels:
  - Culturally biased
  - Often poorly express the purpose of the choice
- Icons are used to suggest purpose
- Example... 
- Icons can be used as, or added to, menu items for the same reason

Example

Format menu

Font icon

Paragraph icon



# Keyboard Input for Menu Navigation

- Besides using a pointing device, most menus support keyboard input
- Best for expert/frequent users (because they are better at *recall*. For novices, *recognize* is better.)
- Typically use function keys or modifier keys (shift, control, alt)
- In many settings, systems are required to support full interaction using only a keyboard for input
  - The goal: **Accessibility** for people with disabilities

# Mnemonics vs. Accelerators

- Two techniques for keyboard menu navigation: **mnemonics** and **accelerators**
  - Mnemonics
    - The full menu hierarchy may be accessed using only the keyboard
    - An underlined single letter serves as the mnemonic
    - *Alt-letter* to initiate mnemonic access
  - Accelerators
    - Shortcuts to bypass the menu hierarchy and directly invoke a menu option



Example

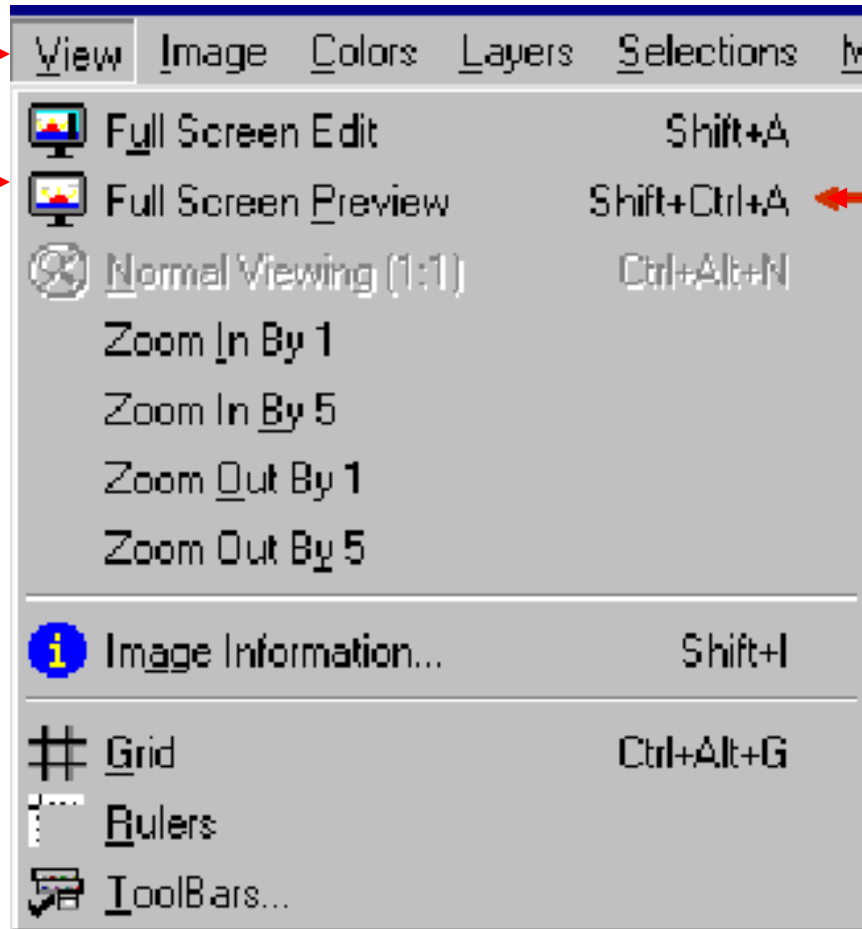
Two keyboard techniques to invoke the “Full Screen Preview” command in the “View” menu

Using mnemonics:

Alt-v

p

View menu



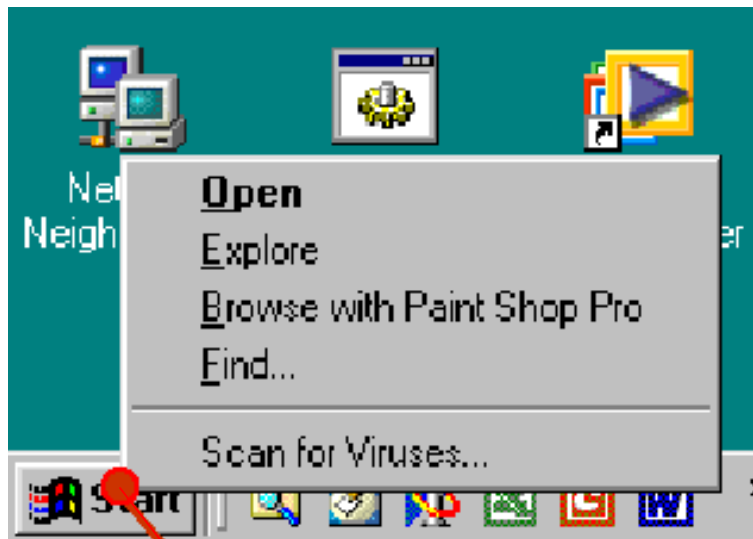
Using accelerator:

Shift-Ctrl-A

# Popup Menu

- Invoked anywhere by right-clicking on mouse button
- Menu that pops up is context sensitive (i.e., depends on where the tracker is when the mouse button is clicked)

Context sensitive popup menus on *Windows* desktop.



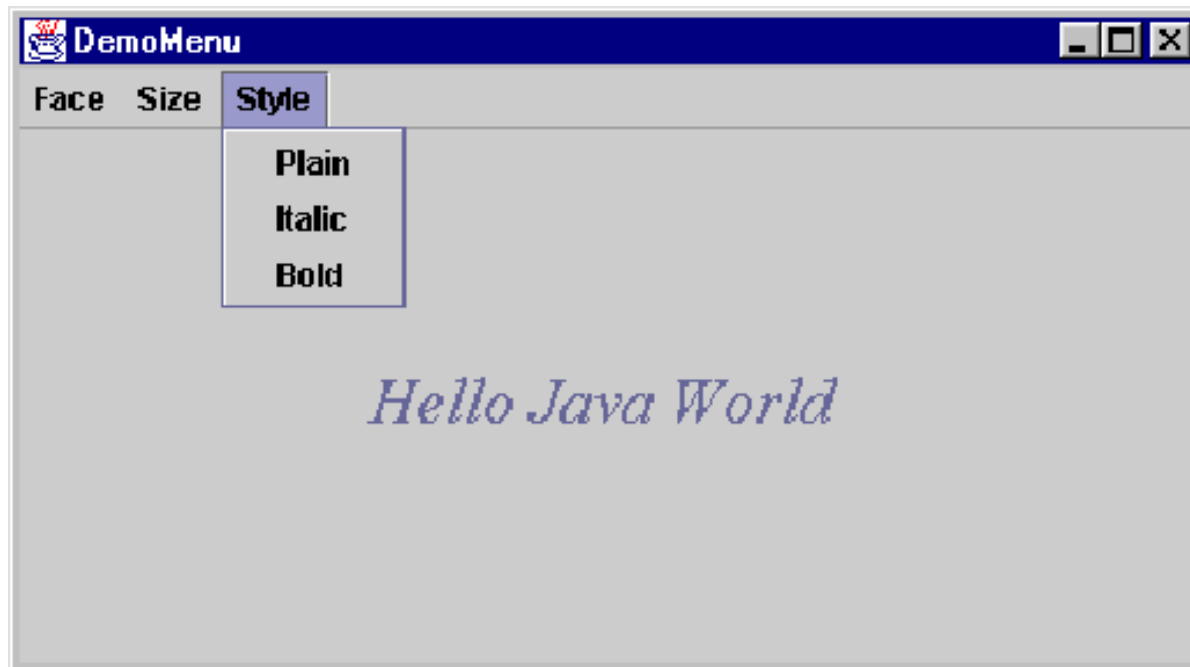
Right-click on  
Start button



Right-click on  
background

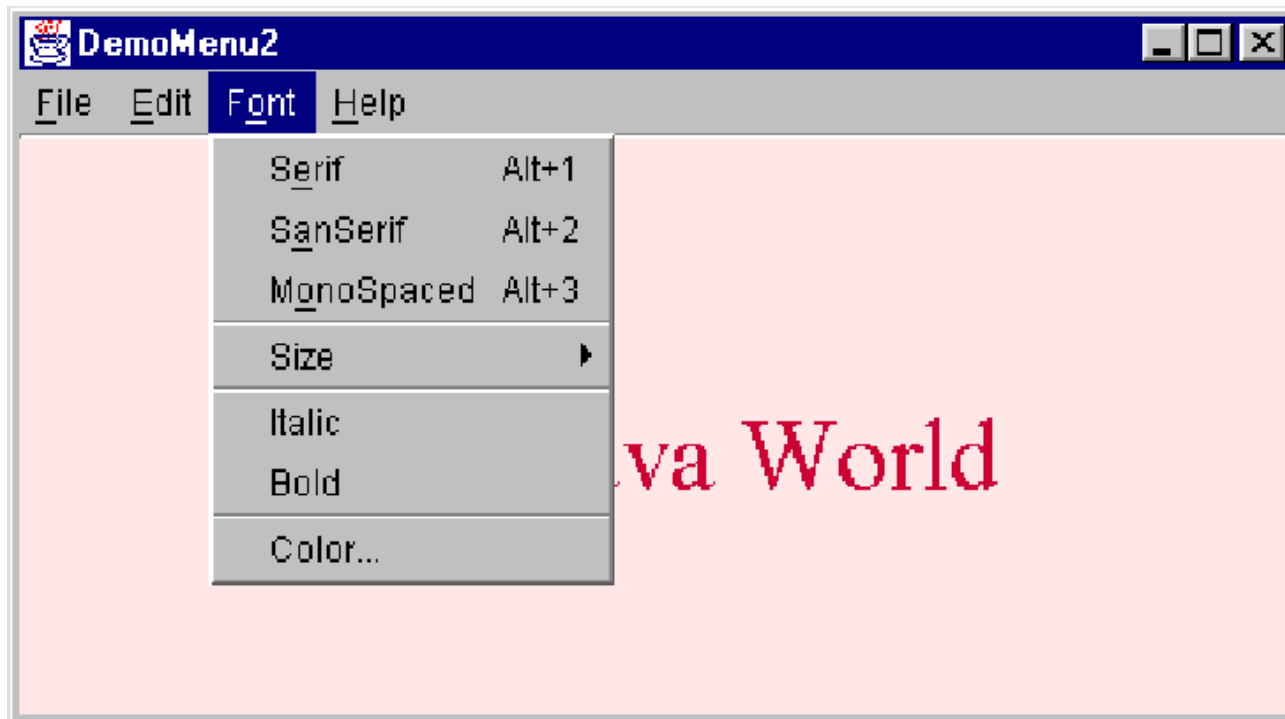
# Example Program

## DemoMenu.java



# Example Program

## DemoMenu2.java

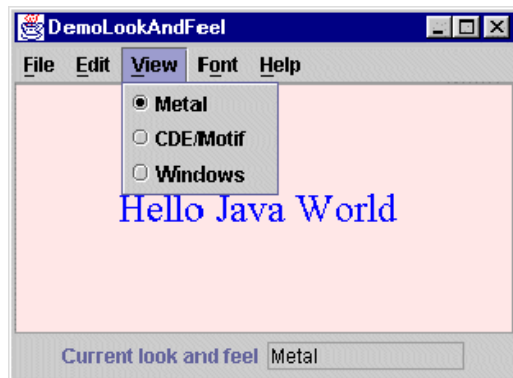


Note: uses Win32 Look and Feel

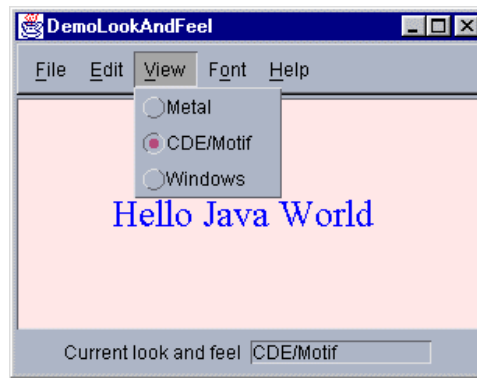
# Example Program

## DemoLookAndFeel.java

Metal (java)



Motif

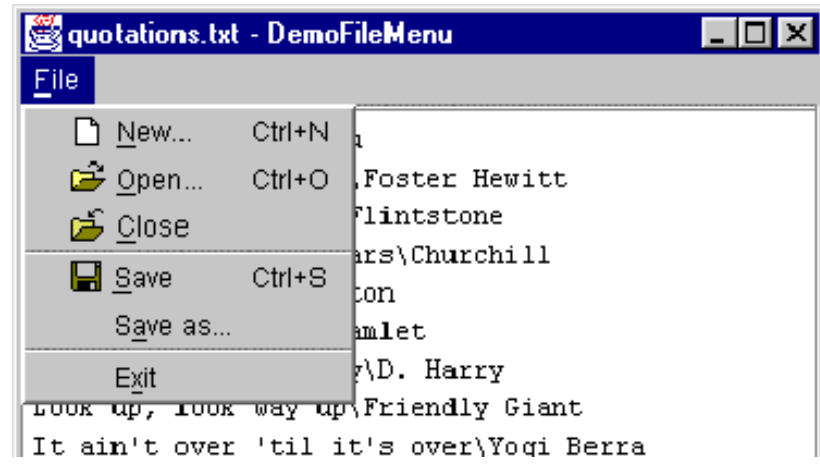
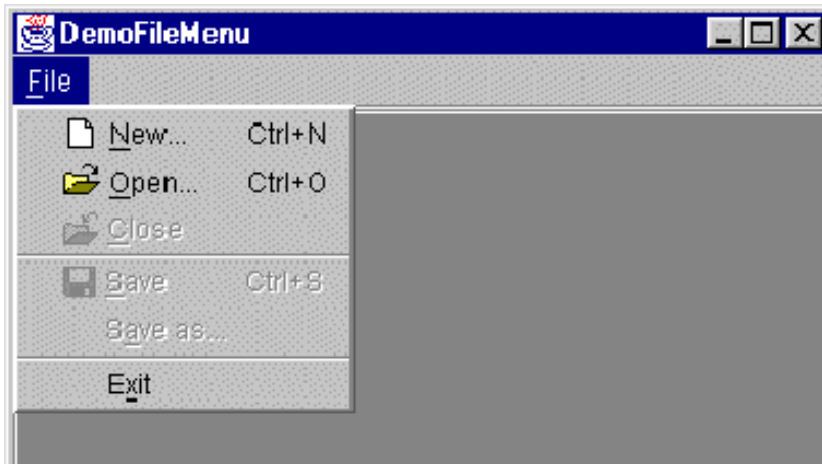


Windows



# Example Program

## DemoFileMenu.java



Next Topic...