

Example Program

DemoLookAndFeel.java



Oct 07

1

Outline

Windows

Icons

Menus

Pointers

Oct 07

2

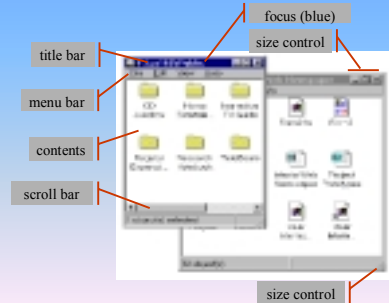
Windows

- 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

Oct 07

3

Parts of a Window



Oct 07

4

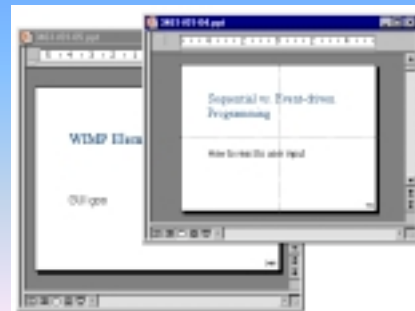
Layout Policy

- 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 ...

Oct 07

5

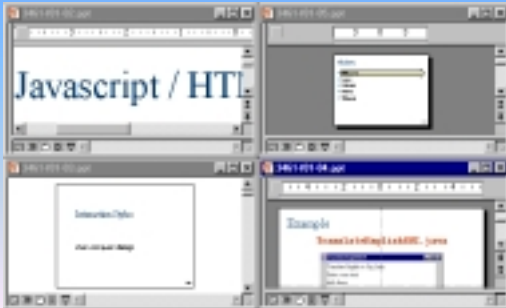
Overlapping Windows



Oct 07

6

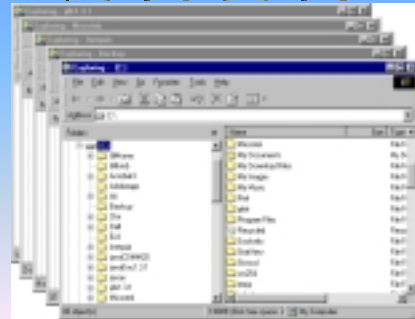
Tiled Windows



Oct 07

7

Cascading Windows



Oct 07

8

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

Oct 07

9

Window Size States

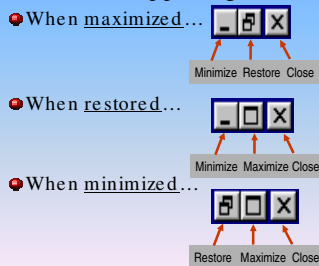
- Windows have three size states
 - Maximized
 - Fills available space
 - Minimized
 - Reduced to a title bar or icon
 - Normal (aka Restored)
 - This is the size of the window, either when it was first opened, or before the window was maximized
 - From this mode, the window width and height may be adjusted

Oct 07

10

Window Size Control

- Via boxes in upper-right corner of window



Oct 07

11

Window Size Control (2)

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



Drag to resize

Oct 07

12

Window Size Control (3)

- Via virtual handles on edges

- When normal...



Drag either edge to resize width

Oct 07

13

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

Oct 07

14

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)

Oct 07

15

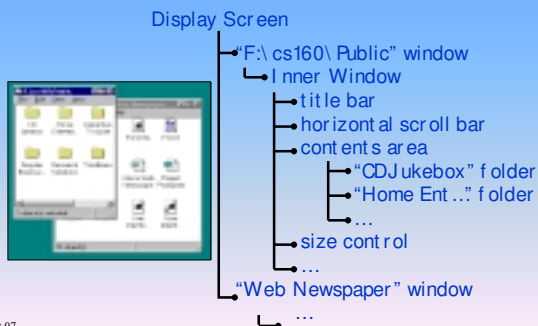
Containment Hierarchy

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

Oct 07

16

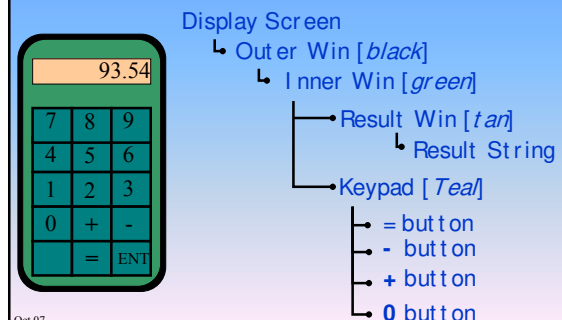
Containment Hierarchy - Example 1



Oct 07

17

Containment Hierarchy - Example 2

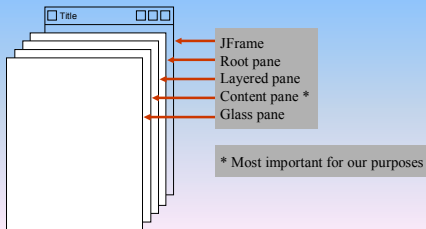


Oct 07

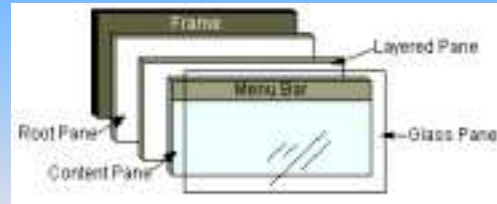
18

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)



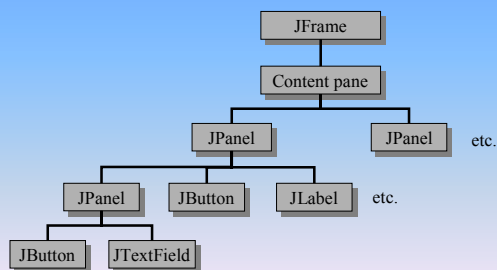
Containers

- 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)
 - 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)
- Oct 07 21

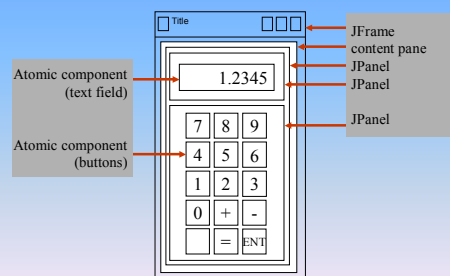
Atomic Components

- An atomic component is a low-level, self-sufficient entity that interacts with the user
 - 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
- Oct 07 22

Containment Hierarchy for JFC/Swing

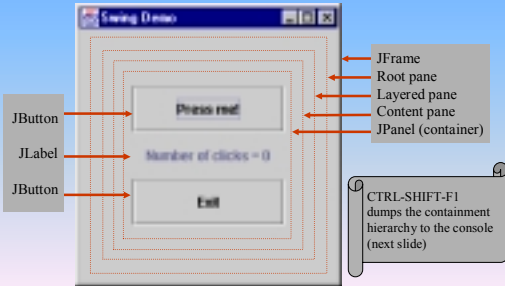


So...



Example Program

DemoSwing.java



Oct 07

25

Containment Hierarchy (abbreviated)

DemoSwing.java

```
DemoSwingFrame[frame0,0,0,224x208, ...
javax.swing.JRootPane[,4,23,216x181, ...
javax.swing.JPanel[null.glassPane, ...
javax.swing.JLayeredPane[null.layeredPane,0,0,216x181, ...
javax.swing.JPanel[null.contentPane,0,0,216x181, ...
javax.swing.JPanel[,0,0,216x181, ...
javax.swing.JButton[,50,50,116x27, ...
javax.swing.JLabel[,50,77,116x27, ...
javax.swing.JButton[,50,104,116x27, ...
```

Oct 07

26

Outline

- Windows
- Icons
- Menus
- Pointers

Oct 07

27

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

Oct 07

28






Advantages of icons

- Save screen space
- Serve as a reminder of available dialogs, applications, or commands that may be restored or invoked

Oct 07

29

Icons Are Used to Represent...

- Disk drives 
- Folders 
- Available applications 
- Files 
- Minimized applications 
- Commands 
- Minimized windows 
- States 

Oct 07

30

Outline

- Windows
- Icons
- Menus
- Pointers

Oct 07

31

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

Oct 07

32

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, etc.
 - Spatial displacement between input device and display
 - Tracker needed
 - Selection via button presses

Oct 07

33

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

Oct 07

34

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

Oct 07

35

Tracker Hot Spot

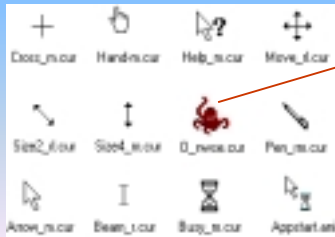
- The tracker is a bit-mapped image (x by y pixels)
- One pixel in the image is defined as the hot spot
- Selection determined by the coordinate of the hot spot
- When designing custom trackers, make sure the image has an obvious hot spot if selection is required while the tracker is displayed

Oct 07

36

Tracker Examples

- Examples from MS Windows



Where is the hot spot?

Oct 07

37

Example Program

`DemoCursorControl.java`



Note:
Cursor is not captured with "print screen". Run the program to observe the cursor changing.

Oct 07

38

More Widgets

Outline

- More text components

- Tool bars
- Sliders
- Scrollbars
- Lists
- Tables

Oct 07

40

Validating Input

- Input data must be in the format required by the application
- Examples:
 - Numeric field with value in certain range
 - Postal or zip code
 - Date/time formats
- Invalid input must be corrected before proceeding
- If invalid input, can present a popup message, generate audio alarm, etc.

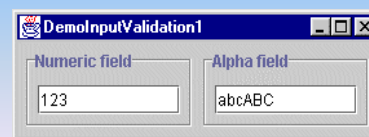
Oct 07

41

Example Program

`DemoInputValidation1.java`

`DemoInputValidation2.java`



Oct 07

42