



## GUI Implementation Support

---

### Learning Objectives:

- Why GUIs?
- What is a GUI?
- Why is implementation support needed?
- What kinds of implementation support are available?
- Basic concepts in OO GUI toolkit & app framework
- Basics of Java Swing
- Use Netbeans to Design a GUI



## Why GUIs?

---

- 1. Feedback**
2. Speak the User's Language
- 3. Clearly Marked Exits**
4. Consistency
5. Prevent Errors
- 6. Minimize User Memory Load**
- 7. Flexibility / Shortcuts**
8. Simple Design
9. Good Error Messages
10. Help and Documentation
11. Use Appropriate Affordances
- 12. Visibility / Obviousness**



## What's a GUI?

### Standard Elements of a GUI

---

- WIMP
  - Windows
  - Icons
  - Menus
  - Pointers



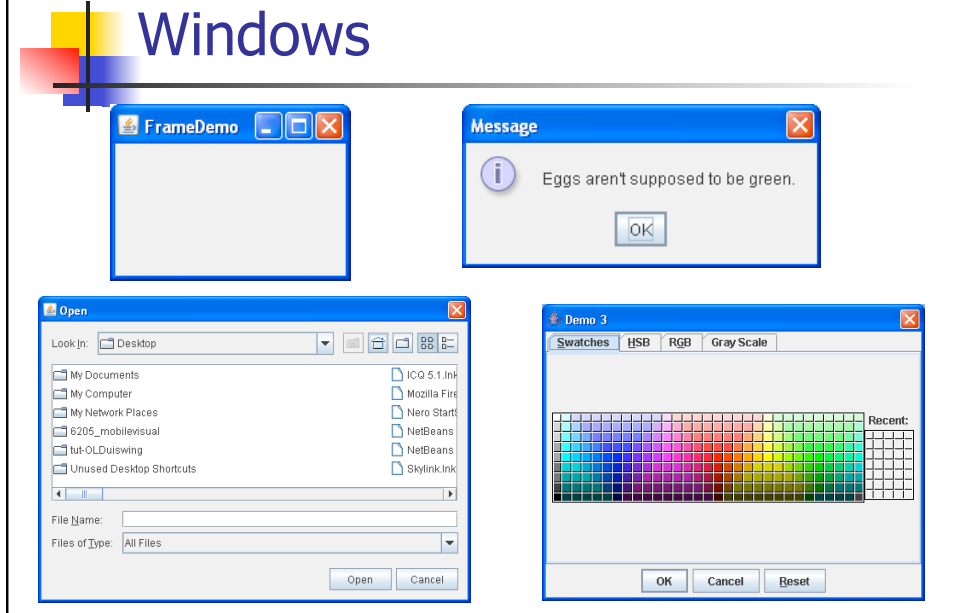
## What is a Widget?

### Examples?

---

- The "Macintosh 7" 1984
  - Button
  - Slider
  - Pulldown menu
  - Check box
  - Radio buttons
  - Text entry fields
  - File pick/save
- *These have become standard*

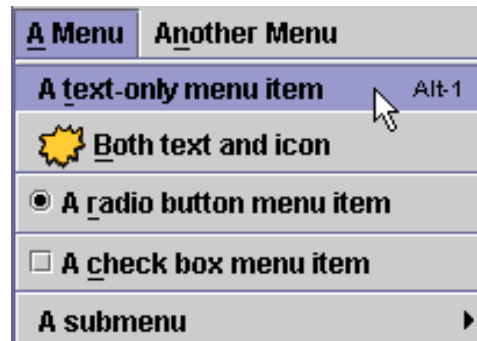
## WIMP Elements in Swing: Windows



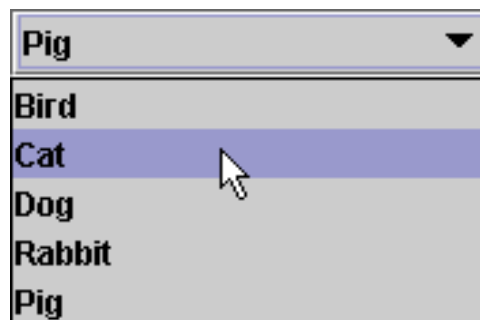
## WIMP Elements in Swing: Buttons



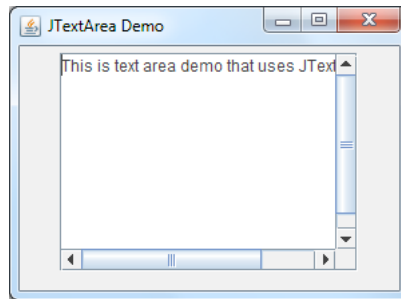
## WIMP Elements in Swing: Menu



## WIMP Elements in Swing: Combo Box



## WIMP Elements in Swing: Text Field / Area



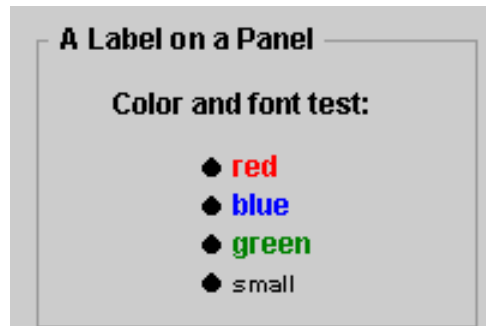
## WIMP Elements in Swing: Labels



## WIMP Elements in Swing: Tool tips



## WIMP Elements in Swing: Embedded Panels



## WIMP Elements in Swing: Advanced

The image displays several standard Swing GUI components:

- A file explorer showing a tree structure: Music (Classical, Beethoven, Brahms, Mozart, Jazz, Rock).
- A table with the following data:
 

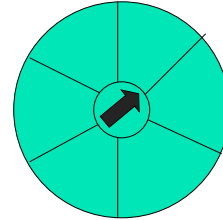
First Name	Last Name	Favorite Food
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	
- A button group with three buttons labeled "One", "Two", and "Three", each with a sun icon.
- A text area containing the text "Blah blah".
- A slider labeled "Frames Per Second" with a range from 0 to 30 and a value of approximately 15.
- A window containing a yellow taxi icon.

## Pros and Cons of "Standard" Widget Sets?

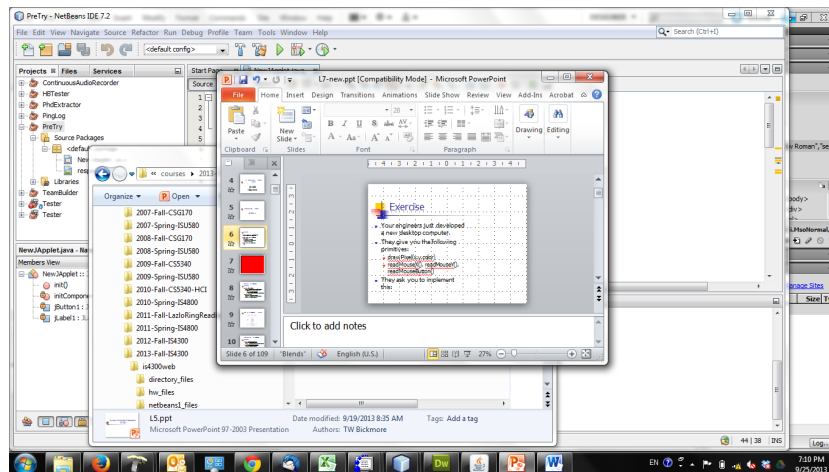
- Pro: Collection of good interaction techniques that work well
  - uniformity is good for usability
  - Improves external consistency
- Cons: Significant stagnation
  - Failing to customize interaction techniques to tasks
  - Efficiency could be improved

## Example of non-standard widget: Pie menu

- A circular pop-up menu
  - only angle of mouse motion counts
  - *Maya, Blender, Grand Theft Auto V, Android Browser*
  
- What are Fitts' law properties?
  - minimum distance to travel
  - minimum required accuracy (dependent on # of options)
  - very fast (dependent on # of options)



## GUI Implementation Support

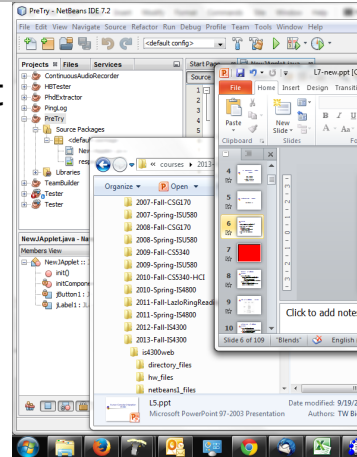




## Exercise



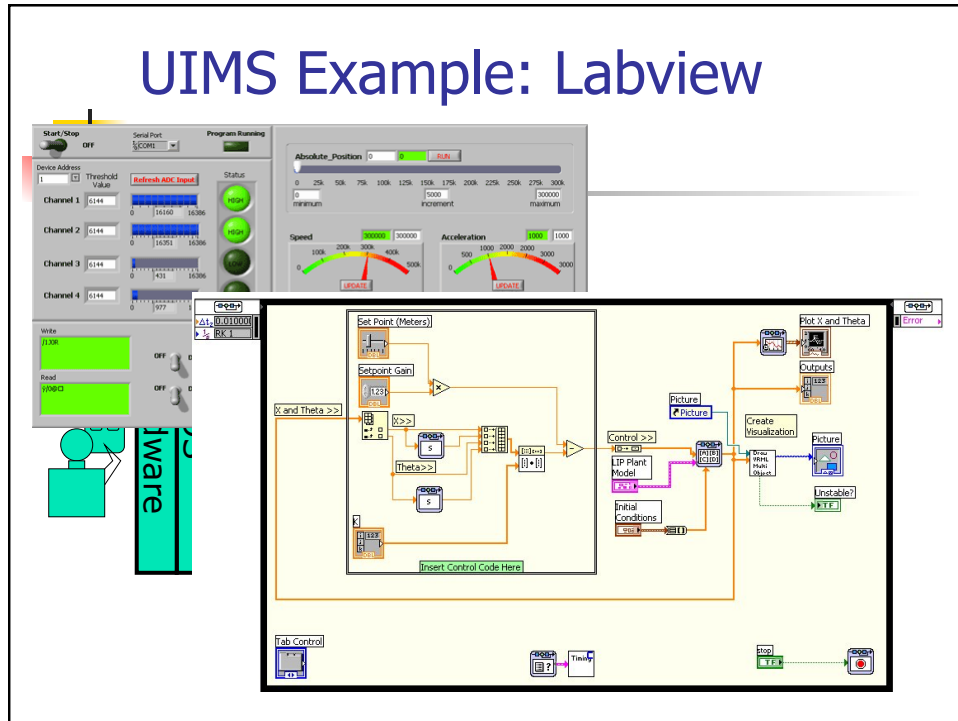
- Your engineers just developed a new desktop computer, and want a modern GUI on it.
- They give you the following primitives:
  - `drawPixel(x,y,color)`
  - `readMouseX()`, `readMouseY()`, `readMouseButton()`, `readKey()`
- What functionality do you need?



## Myers: Levels of Abstraction in UI Software



## UIMS Example: Labview

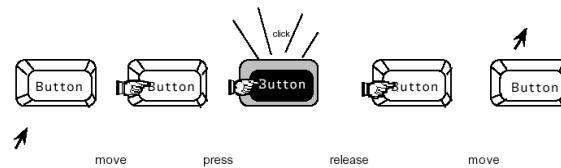


## Myers' Chapter Graphical User Interface Programming

- Why use GUI tools?
  - Makes authoring easier & more economical
  - Improves quality
  - Improves usability

## OOP and Toolkit Widgets

- Why are they so well suited?
  - Natural metaphor (direct manipulation)
  - Subclassing to create custom widgets
  - Encapsulation (data & behavior)



## OO Toolkit Concepts

### #1 Specialization via Subclassing

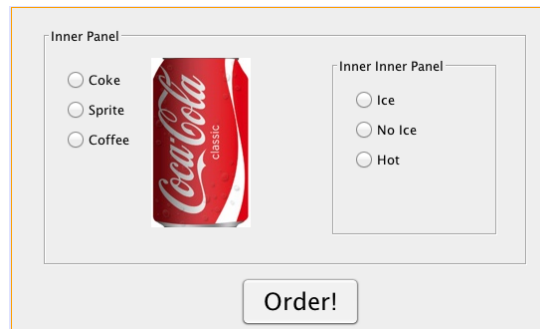
```

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.text.JTextComponent
          javax.swing.JTextField
            javax.swing.JTextArea
  
```

Years:

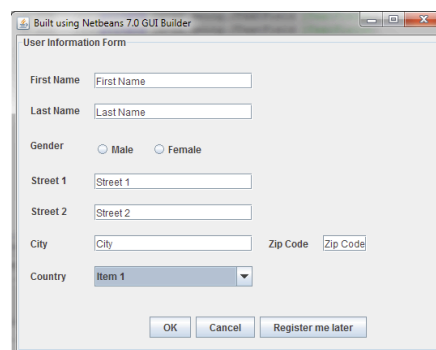
## OO Toolkit Concepts #2 Composition

- Put together interactive objects at larger scale than atomic interactors
- Container objects



## OO Toolkit Concepts #3 Layout

- How a container organizes its widgets within itself.

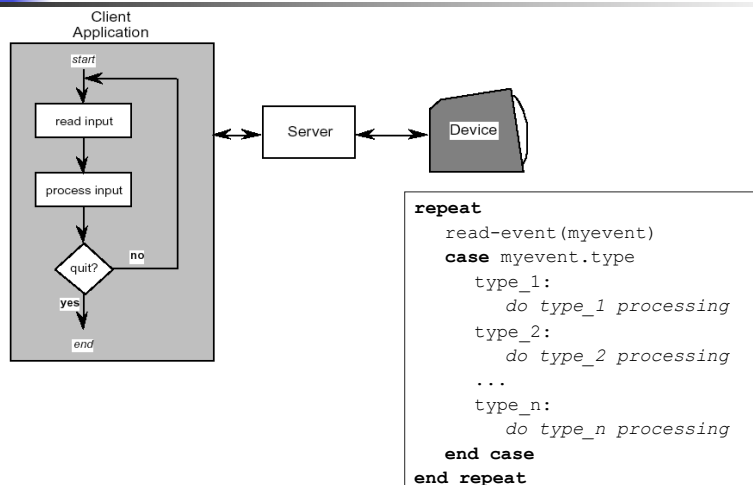


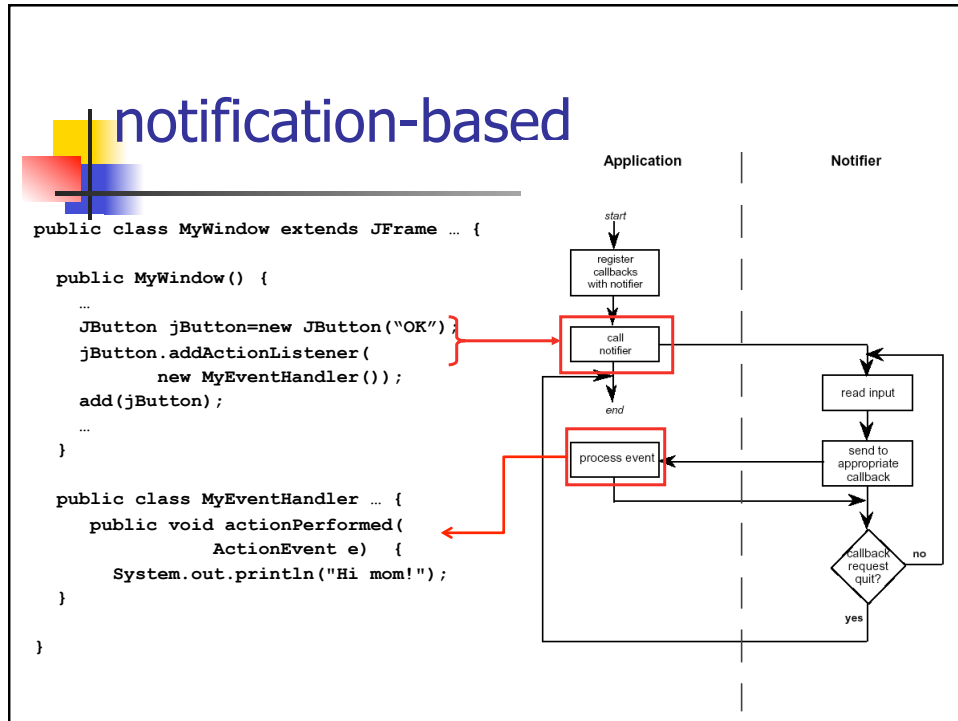
## OO Toolkit Concepts

### #4 Event Handling

1. When anything happens in the UI
  - Mouse clicked, Window moved, Key pressed, etc
2. Windowing System creates a record
3. The event record is added to a UI event queue
4. The application (or toolkit) pulls events from the queue and acts on them in order

## read-evaluation loop

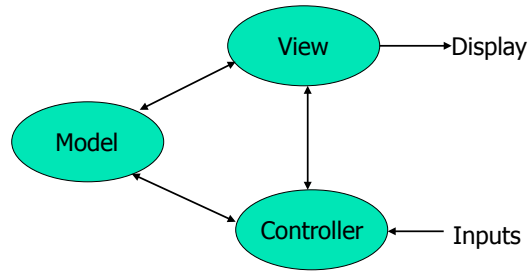




## OO Toolkit Concepts #5 Virtual toolkits & MVC

- Aka Cross-Platform Development Systems
- Provide a layer of abstraction above "native" toolkit
- Two approaches:
  - Map to native widgets
  - Provide own widgets

# Model-View-Controller Architecture for Widgets

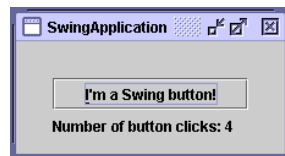


# Swing

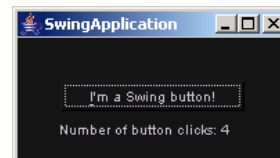
## Java GUI APIs

- AWT
  - The original – now mostly obsolete as a toolkit (event handling mechanism still used in Swing)
  - Used “heavyweight” components
- Swing (~1997)
  - The current(?) standard.
  - Native window, but draws all widgets
  - Pluggable look-and-feel
- SWT (Standard Widget Toolkit)
  - Open source widget toolkit.
- JavaFX
  - Becoming new standard UI toolkit (?)

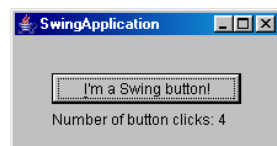
## Pluggable Look-and-Feel



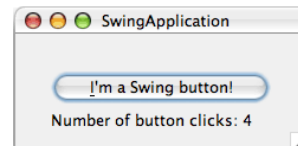
Java



GTK+

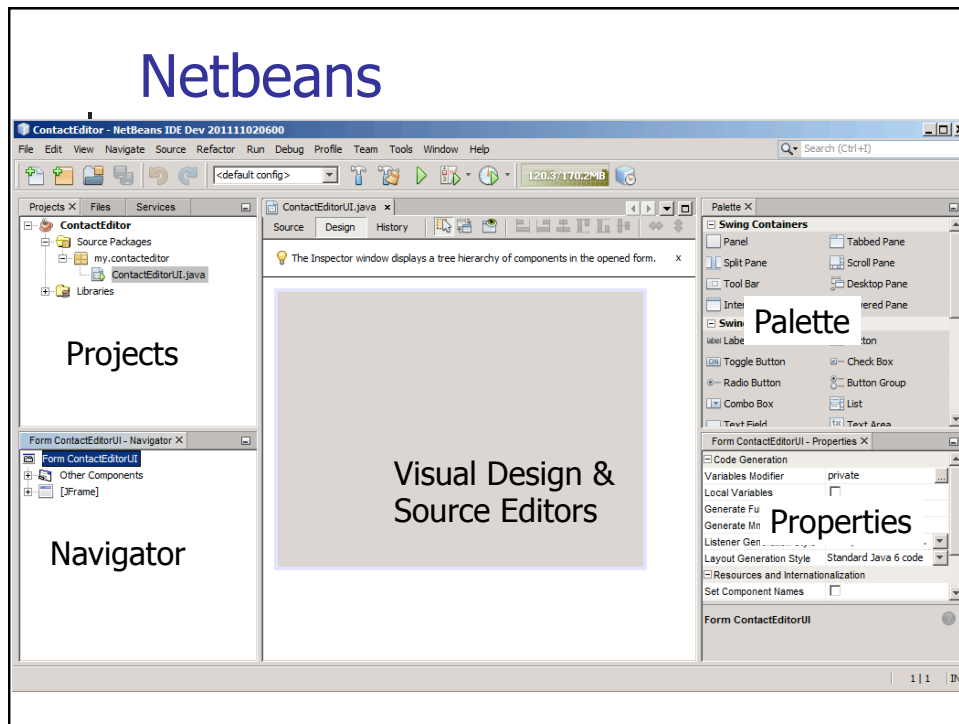


Windows



Mac





## Group Exercise

- Project Teams
- Sketch a "main screen" for your app



## GUI Implementation Support

---

### Learning Objectives:

- Why GUIs?
- What is a GUI?
- Why is implementation support needed?
- What kinds of implementation support are available?
- Basic concepts in OO GUI toolkit
- Basics of Java Swing
- Use Netbeans to Design a GUI



## Swing Homework I4 – Create a Restaurant Ordering App

---

- Two JLabels, one with an icon.
- Two JButtons, one with an icon.
- Etc.
  
- Email a screen shot of your app running, together with a zip archive of your project directory to is4300f15@
  
- Due in 1 week



## To do

---

- Read
  - Design I (Benyon Ch 9; Rosson Ch 3).
  
- Due Monday: P2 – Requirements Analysis
  
- Start Homework I4 – Swing & Netbeans