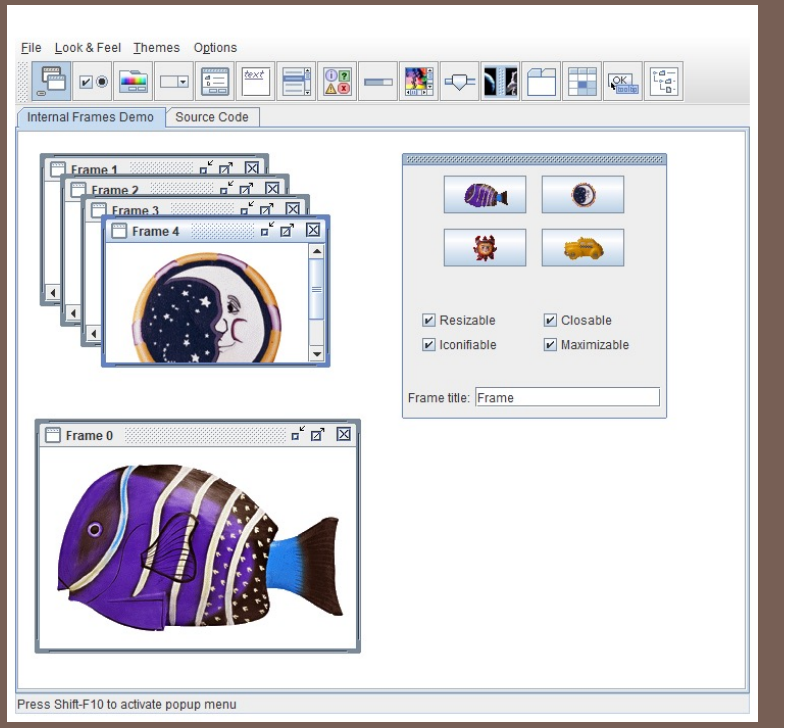
A5 due in 1 week



GRAPHICAL USER INTERFACES, PART 1

Lecture 18

CS2110 Fall 2022

Graphical User Interface (GUI)

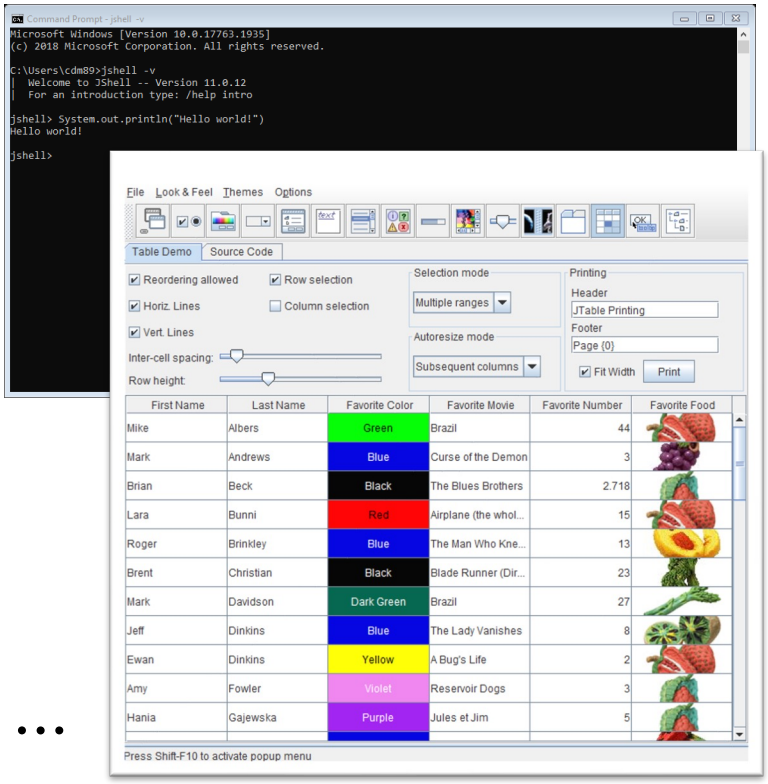
- Enables rich interaction between user and program with easier learning curve
- big motivator for development of OO!
  - WIMP model developed along with Smalltalk, an early OO language
  - "WIMP" = Window, Icon, Menu, Pointer
  - at Xerox PARC in 70's, popularized by Macs

History of (desktop) Java GUI frameworks

- AWT: Abstract Window Toolkit
  - Used "native" (OS-specific) GUI elements, but only lowest-common-denominator of features supported by all platforms
- JFC/Swing (1998)
  - Drawing and events all handled by Java ("lightweight")
  - Builds on AWT abstractions
- SWT (2003)
  - Invented for Eclipse
  - Uses native widgets, but more features than AWT
- JavaFX (2008)
  - Modern alternative to Swing, leverages web technologies
  - Not included with Java by default

Where do inputs come from? Where does output go?

- Batch application
  - Input: command-line arguments, files
  - Output: files
- Interactive command-line application
  - Input: user typing (`System.in`)
  - Output: console (`System.out`)
- Graphical User Interface (GUI)
  - Input: mouse, keyboard, gamepad, …
  - Output: formatted text, images, animations, …



GUI Programming Techniques

- Reactive, event-driven programs; inversion of control
  Code is executed *in response to* mouse clicks, keyboard input, etc.
- Multiple threads of execution
  Can perform background calculations while animating appearance
- Separation of concerns: Model, View, Controller (MVC) pattern
  - Model: application state and logic = familiar Java code, decoupled from…
  - View: Show state to users, accept inputs (this lecture)
  - Controller: Respond to events, update model (next lecture)

Making windows

- `javax.swing.JFrame`: a primary application window
  - Has many properties with getters and setters (> 100 methods)
    - Title
    - Close button
    - Menu bar
    - Content area
    - Visibility
    - Size, location, …

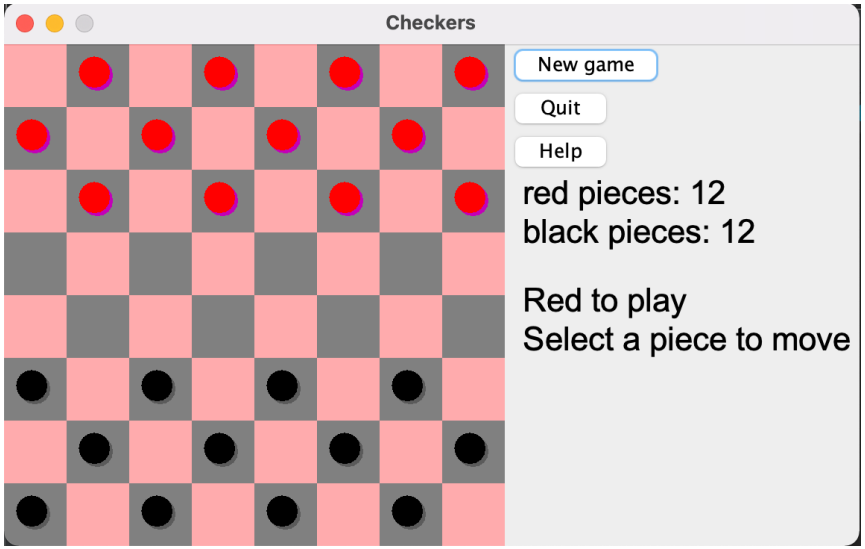
JShell demo

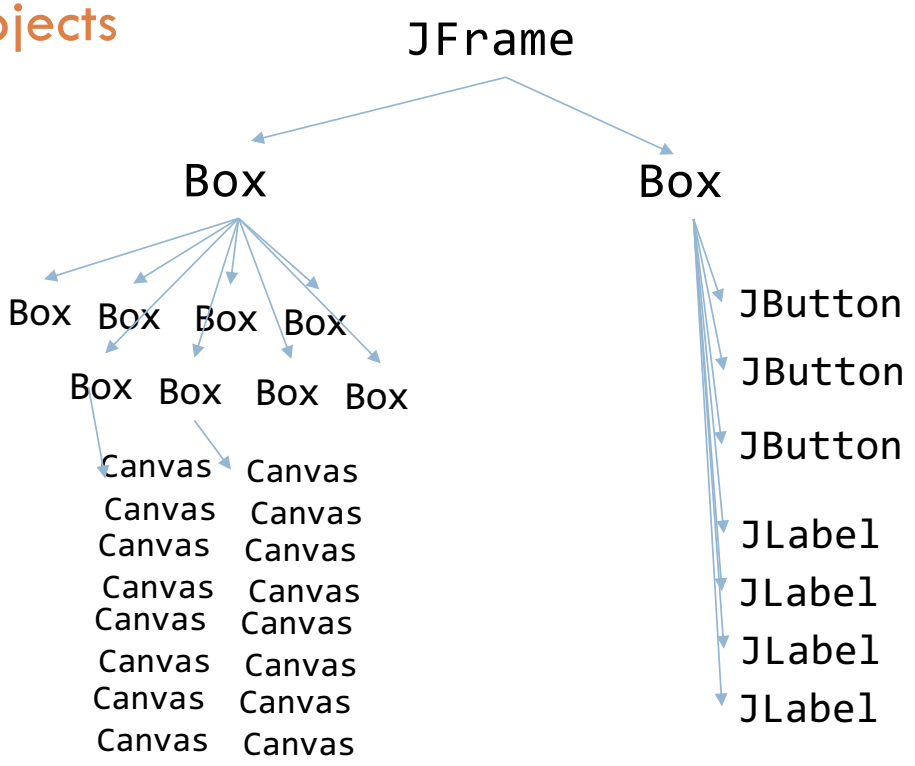
- Note: Using JShell violates an important concurrency rule: only one thread can access Swing component objects
  - But it's safe enough for a demo

1. Make a frame
2. Set its title
3. Specify what happens when close button is clicked
4. Add content
5. Determine optimal size
6. Make visible

Component Hierarchy

- GUI is built from component objects with a tree structure.
  - Root of tree = JFrame

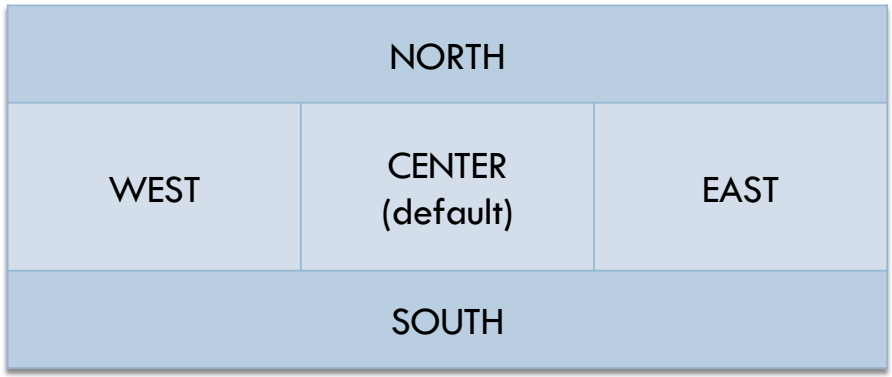




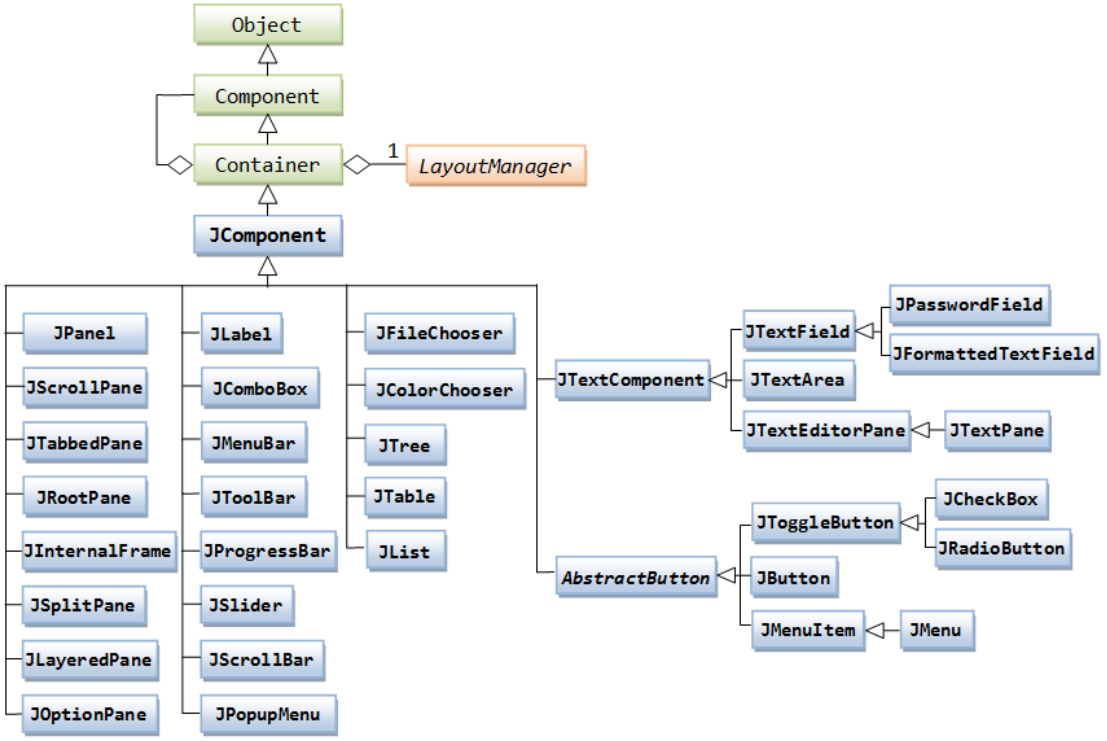
Adding components

- Can `add()` a component to a frame, but where will it go?
  - Governed by a Layout Manager
  - Default for `JFrame`: BorderLayout
- Example:

```
frame.add(
   new JLabel("Hello"),
   BorderLayout.NORTH)
```

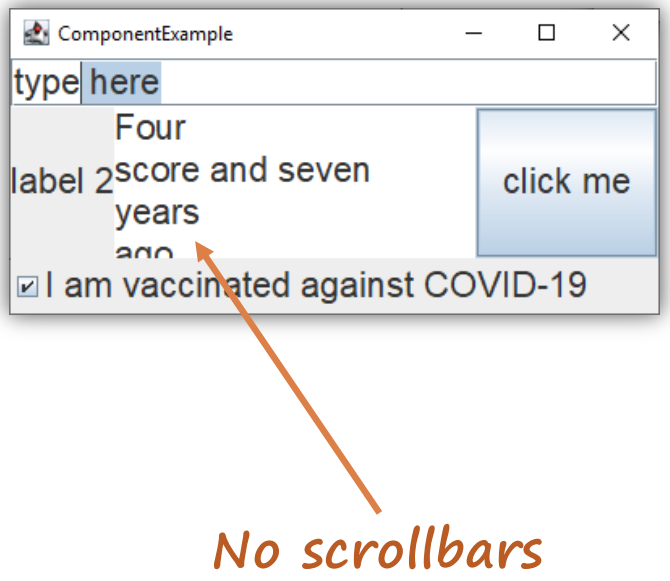


Available components



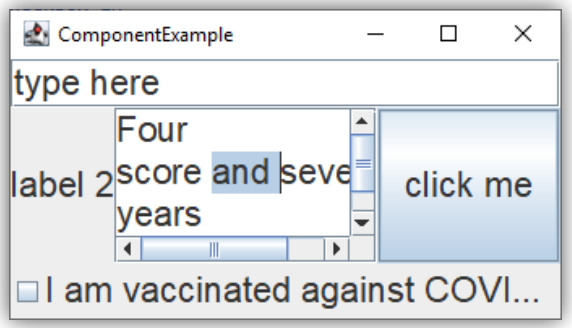
A few common components

- `JTextField`
  - Can recommend width (in characters), but layout manager can overrule
  - Access contents with `getText()`
- `JLabel`
  - Can customize font
- `JTextArea` (multiline)
- `JButton`
- `JCheckBox`

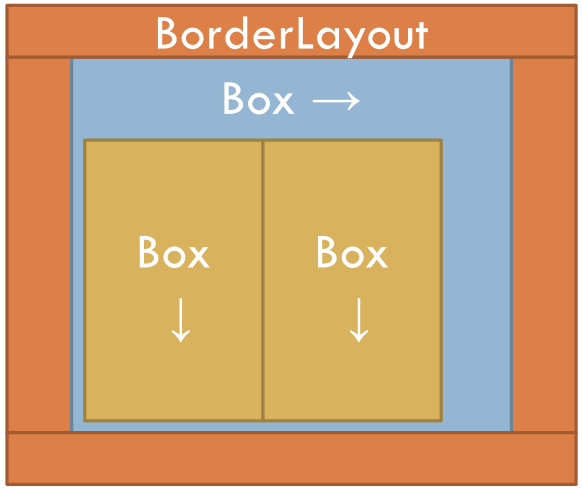


Wrapping components

- `JScrollPane`
  - Construct scroll pane around text area, then add scroll pane to frame (don't also add text area – it is already in the hierarchy)



Demo: BoxLayout



Custom components

- Extend JComponent (or JPanel)
- Override paintComponent(Graphics g) to define appearance
- Attach event listeners
- Graphics, Graphics2D: draw on screen
  - Basics: set color, fill shape
  - When using Swing, can cast g to Graphics2D; many more drawing features
  - Coordinate frame: origin in top left; y increases down

Inversion of control

- Hollywood principle: "Don't call us; we'll call you"
  - User and environment dictate when things happen
- Single-threaded, but not the "main" thread
  - Event loop: waits for events to happen, then executes code in response
  - Avoid race conditions by queueing events to do your work: SwingUtilities.invokeLater()

- When does program exit?
  - main() must return, AND
  - All windows are disposed AND
  - Nothing generating events in background (e.g. timers)

  - OR call System.exit(0) (e.g. EXIT_ON_CLOSE)

Next steps

- Next lecture: listening for and responding to events
  - Button presses, mouse movement, etc.
- Exercise: Build your own interface
  - Consider modifying an existing example
  - Lots of details to pay attention to if you want a polished product
    - Keyboard shortcuts, hotkeys, tooltips, right-click menus, localization
- Browse the Swing Tutorial
- Look at demo classes – very short and focused