Rolling Your Own Components

- JButtons, JLabels, JTextFields, JSliders, JTables, JLists — they’re all nice, but sometimes you have to make your own components

- Read-only Swing components have two primary tasks; editable (interactive) components have a third:
  1. How to display me
  2. How big I want to be
  3. What input I take, and how I pass it on

First Steps

- By “custom component,” we mean custom functionality, not just appearance
  - For look-and-feel (LAF) customization, look at UIManager and pluggable LAF APIs; in JDK 1.5, look up skins
  - …unless it’s sufficiently simple and you won’t need a lot of generality or reusability

- Choose your superclass
  - Most of the time, this is JComponent
  - Sometimes JPanel may make sense, though ideally, you should think about what is truly “custom” about your panel and factor that out into a JComponent which you then stick in a JPanel in the usual way
“How to Display Me”

• One entry point: `public void paintComponent(Graphics)`
• Respect opacity: if `isOpaque()`, the view must cover its entire area in `paintComponent()`
• Respect the border: use `getInsets()` to figure out how much space is used by your border (if any)
• Fancy drawing is best done with a `Graphics2D`; you can derive it from the `Graphics` object using `create()`:
  ```java
  Graphics2D g2 = (Graphics2D)(g.create());
  ```

“How Big I Want to Be”

• Main override: `public Dimension getPreferredSize()`
• As with drawing, respect the border: use the values in `getInsets()` when you calculate your preferred size
• If you want to express a minimum or maximum size, then also override `getMinimumSize()` and `getMaximumSize()`
• Careful when tweaking `getPreferredSize()` for a JPanel: the base implementation includes code that recursively considers the sizes of the panel’s children
What Input I Take, and How I Pass It On

• Two layers of events:
  ◆ Low-level events — “close to the iron,” such as mouse events, keyboard events
  ◆ Higher-level (some might say “semantic”) events — more conceptual, carries the meaning of an event instead of its mechanism (e.g. ActionListener, ChangeListener, ListSelectionListener)

• A typical custom component receives low-level events, interprets them, then relays them as higher-level events

Input Alternatives

These fall into the category of “if you don’t really need to do all that work…”

• No event firing: custom component doesn’t inform listeners of any changes; callers must use getters and setters to read/change the state of the component

• Event relaying: custom component doesn’t really deal with low-level events; instead, it takes low-level listeners too (MouseListener, KeyListener) and just passes mouse and key events to those listeners