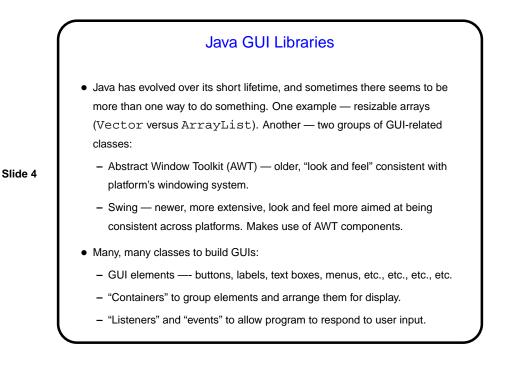
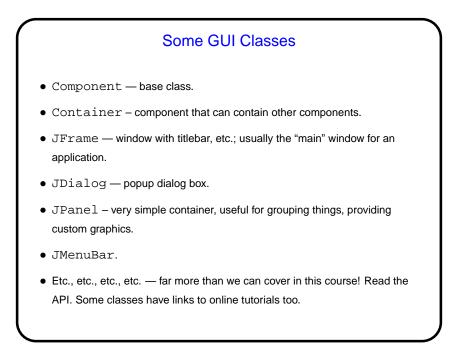




- In PAD I (and in most previous in-class examples this semester) we usually focus on programs with simple text-based input and output — a basically synchronous interaction with the user.
- Programs with GUIs, though, are typically somewhat different the main program (which is sometimes hidden in library code) is often just a loop that waits for keyboard/mouse input delivered by the program's environment (operating system, graphical environment, window manager, etc.).
- This leads to an "event-driven" programming model that can seem rather different from what's used for text-based programs. (But it's rather like what you're doing in the game project.)





Using the GUI Classes — Appearance

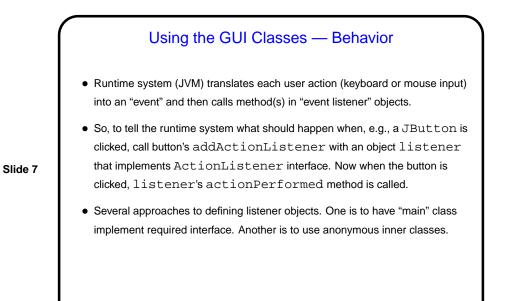
• When using predefined components, key issue is how they're grouped using containers and how things are laid out within each container.

 Preferred method for laying things out — layout manager, which places elements in some reasonable way, does something reasonable if container is resized.

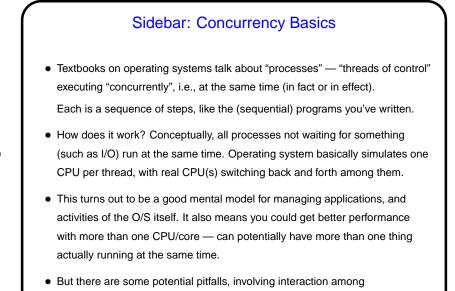
- Simple layouts include FlowLayout, GridLayout, BorderLayout, BoxLayout.
- GridBagLayout provides more control, but is more complex.

Some of them expand components to fit, others lay them out at their minimum size. See API and tutorials for more info.

 Often makes sense to group elements hierarchically — JPanel is useful for that.



Examples
(Examples as time permits.)
Before going further, we need what seems like a detour ...



processes/threads.

Sidebar Continued
Two basic models — one in which the concurrently-executing things don't share (much) memory and one in which they do. Sharing memory has benefits but also some serious potential pitfalls ("race conditions").
Java provides some support for both models, but at this point its support for the shared-memory model is more relevant, because ... (to be continued).

