Event loops

Programming in prehistoric times
- Earliest programs were all “batch” processing
- There was no interaction with the user

Very early interactive programs
- BASIC was an early interactive language
- Still a central computer, with terminals
- Style of interaction was “filling out forms”

Command-driven programs (20 years ago)
- Allow the user to enter “commands”
- Much more flexible
- Still only a single source of inputs
- Not good enough for modern programs

Input
Process
Output

Ask user for command
Read and parse command
Execute command
quit

Ask user for input
Process input
Ask user for input
Process input
etc.
Modern event-driven programs

- The rise of Graphical User Interfaces forced a change
- Multiple sources of input
  - mouse clicks
  - keyboard
  - timers
  - external events
- A new program structure is required
  - Also useful for programming distributed system components (e.g., Internet programs) too

Java hides the event loop

- The event loop is built into Java GUIs
  - GUI stands for Graphical User Interface
- Interacting with a GUI component (such as a button) causes an event to occur
- An Event is an object
- You create Listeners for interesting events
- The Listener gets the Event as a parameter

Building a GUI

- To build a GUI in Java,
  - Create some Components
  - Use a layout manager to arrange the Components in a window
  - Add Listeners, usually one per Component
  - Put code in the Listeners to do whatever it is you want done
- That's it!
  - Of course, there are a lot of details....

Vocabulary I

- event – an object representing an external happening that can be observed by the program
- event-driven programming – A style of programming where the main thing the program does is respond to events
- event loop – a loop that waits for an event to occur, then dispatches it to the appropriate code
- GUI – a Graphical User Interface (user interacts with the program via things on the screen)
Vocabulary

- Component – an interface element, such as a Button or a TextField
- Layout Manager – an object (provided by Java) that arranges your Components in a window
- Listener – an object you create to execute some code appropriate when an Event occurs