GUIS

GUI BREAKDOWN

- Composed of 3 objects
  - Components
  - Events
  - Listeners

COMPONENTS

- Defines screen element
- Used to display info + allow users to interact with program
- Examples:
  - buttons
  - text fields
  - labels
  - scroll bars
- Containers = special component to hold/organize other components

EVENTS

- Object that represents some occurence
- Often correspond to actioins like mouse click, typing
- components generate events to indicate user action
- Examples:
  - button generates event to indicate pushed
- Event-drive -> program oriented around GUI responding to user events

LISTENERS

- Object that waits for events to occur
- Responds when event occurs

TYPICAL GUI

- Typically use predefined components and events
- Typically write listener classes to perform desired actions
- GUI Steps:
  - instantiate and setup components
  - implement listener class that specifies what to do when event occurs
  - specify relationship between components and listeners
- Mainly in `java.awt` and `javax.swing`

SPECIFICS - FRAME

- Container to display GUI app (window / title bar)
- JFrame class
- Heavyweight -> managed by OS

SPECIFICS - PANEL

- Container that cannot be displayed separately
- Must be a part of another container
- Role -> organize components
- JPanel class
- Lightweight -> managed by program

SPECIFICS

- Labels -> `JLabel` (display text, image, etc)
- Buttons -> `JButton` (push generates action event)
- Checkbox -> `JCheckBox`
- Radio button -> `JRadioButton`
- Slider selection -> `JSlider` (drag left-right to select number)
- Drop-down selection -> `JComboBox`
- `Timer` (in `javax.swing`) -> no visual representation, simply generates action event at specified interval

LAYOUT MANAGERS

- determine how components in container are arranged
- do not need to specify
  - default is FlowLayout
- Set by:

```
JPanel panel = new JPanel();
panel.setLayout(new GridLayout())
```

LAYOUT MANAGERS - CHOICES

- `BorderLayout` -> 5 areas (N, S, E, W, center)
- `BoxLayout` -> single row/column
- `CardLayout` -> only one component visible at a time
- `FlowLayout` -> left-to-right, add rows as needed
- `GridLayout` -> grid of rows, columns
- `GridBagLayout` -> like grid, but components can span multiple cells
- **See book (ch06) for details on each or Google "Java Visual Guide to Layout Managers"**

MOUSE EVENTS

- listen by implementing `MouseListener` and/or `MouseMotionListener`
- Events:
  - mousePressed, mouseClicked, mouseReleased, mouseEntered, mouseExited (`MouseListener`)
  - mouseDragged, mouseMoved (`MouseMotionListener`)
- Often don't care about all
  - need to implement empty for each

KEY EVENTS

- generated when keyboard key is pressed
  - program responds immediately (no need to press enter)
- listen by implementing `KeyListener`
- Events:
  - keyPressed, keyReleased, keyTyped

MISC

- Dialog Boxes
  - basically popups
  - use `JOptionPane` and its methods:
    - `showInputDialog`
    - `showMessageDialog`
    - `showConfirmDialog`

MISC

- File Choosers
  - `JFileChooser`
  - special dialog box to let user select file
- Color Choosers
  - `JColorChooser`
  - special dialog box to let user pick color
- Other enhancements exist - font changes, borders, tool tips