Slide 1

Administrivia

- Reminder: Homework 4 code due today. Homework 5 due dates posted (next week).
- Reminder: Quiz 4 Thursday. Likely topic is linked lists.

Slide 2

Java GUI Libraries — Recap

- Many, many classes for GUI components — pre-defined components (e.g., `JButton`), containers (e.g., `JPanel`).
- How things are arranged on screen is controlled by “layout manager”. Can nest containers, giving them different layout managers.
- How things work depends on “event listener” methods. Good place to use anonymous inner classes.

Slide 3

Java GUI Libraries — Design Tips

- Probably better not to mix AWT and Swing unless necessary (e.g., unless you're doing an AWT-only program, prefer `JFrame` to `Frame`).
- To find out how to use components — skim online API, Sun tutorials (follow links from API), look for examples similar to what you want to do.
- For small programs, okay to put GUI and underlying data all in one class. For larger programs, consider separating them — "Model/View/Controller" design pattern.
- GUI components that must be accessed by more than one method — e.g., by listener methods — should be instance variables. Other components can often be declared locally in constructor.

Slide 4

Graphics in Java — Custom Components

- Predefined components (`JButton`, etc.) do a lot, but what if you want something that's not provided? in particular, you want to control the image yourself?
- Make a custom component — define a subclass of a component that provides some of the needed functionality, and override the method that defines what's displayed.

  E.g., subclass `JPanel` and override `paintComponent`, to include your code to "paint" the panel.
- Call `repaint` when ready to redisplay.

Slide 5

Custom Painting

- Method to override is `public void paintComponent(Graphics g)`. `g` is a "graphics context" that you can draw on. (Actually it's a `Graphics2D`.) Tutorial recommends calling `super.paintComponent(g)` before doing anything else.
- Can get dimensions of panel with `getSize`, `getHeight`, `getWidth`, `getInsets`.
- Can set colors, draw shapes, lines, text, etc., etc. — see `Graphics` and `Graphics2D` class. Coordinate system is similar to what you're using in your game. See code in `BasicBlock` for simple example.

Slide 6

Custom Painting, Continued

- General advice — look over the methods of `Graphics` and `Graphics2D`; if confused, follow links to tutorial(s) and look for a suitable example to adapt.
- Let's look at example(s) . . .

Slide 7

Drawing and Filling Shapes

- "Draw" means draw outline only; "fill" to draw and fill.
- `Graphics` provides methods for doing simple shapes. `Graphics2D` provides more general methods. (Look at some shapes in `java.awt.geom`.)
- You already know (from your game) about simple way to control color of what's painted. The `Graphics2D` class provides a lot more options (next slide).

Slide 8

Drawing and Filling Shapes, Continued

- `Graphics2D` provides, among other things:
  - `setPaint` to fill shapes with simple color, gradient fill, etc.
  - `setStroke` to draw outlines with different widths, etc.
  - `setFont` to draw text in different fonts. (This works for text components such as `JLabel` too.)
- And there's more — "clipping", affine transformations (e.g., rotation — transformations in which parallel lines stay parallel), etc., etc.
- (Examples as time permits.)

Minute Essay

- None — sign in.