Slide 1

Administrivia

- Reminder: Homework 5 design due today, code Tuesday.
- Reminder: Quiz 4 Tuesday. Likely topic is linked lists.
- Example from last time on Web; other GUI examples we didn't do in class (including one with a `JDialog`).

Slide 2

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 3

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 4

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 5

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 6

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.)

Slide 7

Minute Essay

- In the example shown in class (`ShowImageFromFile` on sample programs page), what would you put in `paintComponent` in `ImagePanel` to fill the panel with a white rectangle with an outline of a red rectangle inside?

Slide 8

Minute Essay Answer

- You could do something like the following:

```
g.setColor(Color.white);
g.fillRect(0, 0, getWidth(), getHeight());
g.setColor(Color.red);
g.drawRect(20, 20, getWidth()-40, getHeight()-40);
```