CS 251
Intermediate Programming
GUIs: Custom Painting

Brooke Chenoweth

University of New Mexico

Fall 2025

Customizing Panels

If you have a `JPanel`, it is simple enough to change its appearance with methods from `JComponent`.

```
JPanel panel = new JPanel();
panel.setBackground(Color.BLACK);
panel.setBorder(
  BorderFactory.createLineBorder(Color.GREEN));
```

Colors

- The `java.awt.Color` class represents colors.
- Contains constants for many default colors.
- Can define your own colors using RGB values.

Borders

- Place a `Border` on a `JPanel` or `JLabel` to specify how to draw the edges of the component.
- Use `BorderFactory` to create a `Border` object.
- Borders can provide lines, titles, empty space, etc.
- Get really fancy with compound borders!

Custom Painting

If you need specify control over graphics, you may need to override the `paintComponent` method of your `JComponent`.

```
public void paintComponent(Graphics g) {
  super.paintComponent(g);

  g.setColor(Color.RED);
  g.drawRect(10, 100, 30, 20);
}
```

Coordinate systems

The Java 2D API uses two coordinate spaces.

User space Device-independent logical coordinate system used by your program.

Device space Device-dependent coordinate system that varies according to the target rendering device.

Coordinate systems

The Java 2D API uses two coordinate spaces.

User space Device-independent logical coordinate system used by your program.

- Top left corner of component's drawing area is (0,0).
- X coordinate increases to the right.
- Y coordinate increases downward.

Device space Device-dependent coordinate system that varies according to the target rendering device.

Coordinate systems

The Java 2D API uses two coordinate spaces.

User space Device-independent logical coordinate system used by your program.

Device space Device-dependent coordinate system that varies according to the target rendering device.

- Coordinate system for a screen may be very different from system used by a printer.
- Conversions between user space and device space are performed automatically.

Graphics methods

- `drawString` For drawing text
- `drawImage` Drawing images (loaded from a file, perhaps)
- Drawing or filling geometric shapes
  - `drawRect`
  - `drawLine`
  - `fillOval`
  - `fillPolygon`
  - etc.

Graphics2D

- The `Graphics2D` class extends `Graphics` with more sophisticated control.
- This is the fundamental class for rendering 2D shapes, text, and images in Java.
- Some methods will give you a `Graphics` object, which you will have to cast to `Graphics2D` to use its methods.

```
public void paintComponent(Graphics g) {
  Graphics2D g2 = (Graphics2D) g;
  // ... do fancy graphics here
}
```