Graphic Interface Programming II

GlassPane, Layered Panes, and animation transformations…
Going outside of the Frame…

- Swing components initially very flexible
- Sometimes you need more
  - Painting over Swing components
  - Advanced animations
  - Non-interruptable events
  - Custom progressbars
  - …

- This can be done with Panes (not Panels)
GlassPane

- The topmost of four layers
  - Like an invisible glass plate over the Frame
Root Pane

- Four parts:
  - Glass Pane – normally invisible top layer
  - LayeredPane - position its contents, which consist of the content pane and the optional menu bar
  - ContentPane – contains all visible components apart from menu bar
  - The MenuBar – home for the root pane's container's menus

- Glass Pane is a Component
Glass Pane

- Useful for custom painting and Animation
- Intercepts (blocks) MouseEvents
- Paints over the application window
- Useful for animations
  - Consider help systems (assignment 4)

- Needs to be made visible before usage
Glass Pane

- Built as a JComponent or a JPanel
  - JComponent is transparent by default
  - JPanel is not transparent

- Can be used to implement Sheet-dialogs
  - PopUp Dialogs that are anchored to a certain Frame
Intercepting Mousevents

// Constructor...
//
public NullEventsGlassPane() {
    addMouseListener(new MouseAdapter() { });
    addMouseMotionListener(new MouseMotionAdapter() { });
    addKeyListener(new KeyAdapter() { });
}
Layered Panes

- Glass Panes are useful...
- One problem:
  - There can only be ONE glasspane

- Layered Panes can be and can be rearranged

- A Pane is a JComponent
Usage?

- Adding images on top of Swing Components
  - Warnings
Example

- Make the input form as usual using standard fields
  - Make the fields detect invalid entries

- Add a Layered Pane that checks for the fields that have detected invalid entries

- Paint the icon in the appropriate places on those fields
Affine Transformations

- Changes relationships between coordinate systems
  - Drawing and Display coordinates, e.g.
- Many operations:
  - Scaling
  - Rotating
  - Moving
  - Shearing
  - …
Examples

Graphics2D g = (Graphics2D) graphics;
...
g.rotate(30.0 * Math.PI / 180.0);
g.setFont(new Font("Serif", Font.BOLD, 24));
g.drawString("M", 0, 0);

Note that the drawing operations are made as if the transformation was not made.

The rotation is made by the rotation transformation
Examples

g.translate(25.0, 0.0);
g.scale(2.0, 2.0);
g.shear(0.0, 0.35);
Example

Graphics2D g = (Graphics2D)graphics;
g.scale(2.5, -1.5);
g.translate(-10.0, 0.0);
g.shear(0.5, 0.15);
g.rotate(10.0 * Math.PI / 180.0);
g.setFont(new Font("Serif", Font.BOLD, 24));
g.drawString("M", 0, 0);

As can be seen here transformations can be combined in any conceivable way
Transformation Ordering

g.translate(25.0, 0.0);
g.rotate(60.0 * Math.PI / 180.0);


g.rotate(60.0 * Math.PI * 180.0);
g.translate(25.0, 0.0);
Example

g.rotate(30.0 * Math.PI / 180.0);
g.scale(2.0, 2.0);
g.translate(0.0, 20.0);
g.scale(3.0, 1.0);
Example

```
g.scale(3.0, 1.0);
g.rotate(90.0 * Math.PI / 180.0);
g.scale(3.0, 1.0);
```
Example

```javascript
// Example 1
g.scale(3.0, 1.0);

// Example 2
g.scale(3.0, 1.0);
g.rotate(25.0 * Math.PI / 180.0);

// Example 3
\[ g.\text{scale}(3.0, 1.0); \]
\[ g.\text{rotate}(90.0 \times \text{Math.PI} / 180.0); \]
```
Important!

class MComponent extends JComponent {

    public void paintComponent(Graphics gIn) {

        Graphics2D g = (Graphics2D) gIn.create();

        g.translate(getWidth() / 2, getHeight() / 2);
        g.rotate(30.0 * Math.PI / 180.0);
        g.setFont(new Font("Sans", Font.BOLD, 24));
        g.drawText("M");

        g.destroy();
    }
}

Important!

class MComponent extends JComponent {

    public void paintComponent(Graphics gIn) {

        Graphics2D g = (Graphics2D) gIn.create();

        g.translate(getWidth() / 2, getHeight() / 2);
        g.rotate(30.0 * Math.PI / 180.0);
        g.setFont(new Font("Sans", Font.BOLD, 24));
        g.drawText("M");

        g.destroy();

    }

}
Global to Local

Point \( p = \text{component.getLocationOnScreen}() \);
\text{SwingUtilities.convertPointFromScreen}(p, \text{this});
Reflected image (example)
ReflectionPanel

- The reflection panel takes an image
- Shows it with a reflection added to it
- If well programmed it can be reused for any image
- A standard component that can be used everywhere
Alpha Channel

- The Alpha Channel decides the degree of transparency
- Using a Gradient Paint is another key to this effect
- The images are composed on the panel
- Combinations with scaling etc can make great looking effects.