1.00 Lecture 17

Introduction to Swing

Reading for next time: Big Java: sections 12.1-12.6

Swing

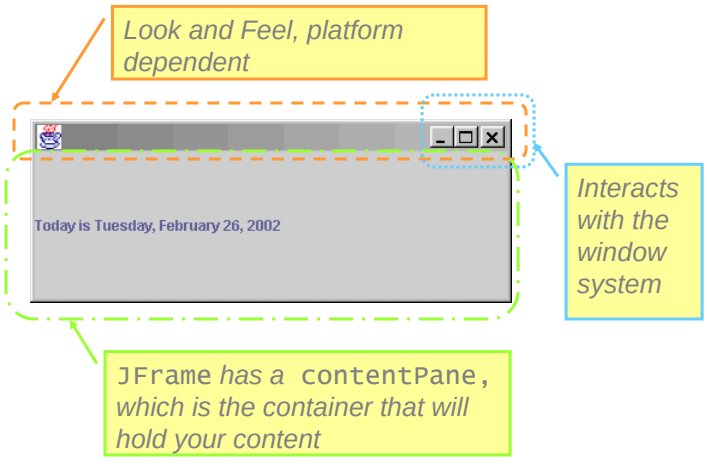
- **Package of user interface classes for windows, menus, scroll bars, buttons, etc.**
- **Independent of hardware and operating system (as long as they can paint a window)**
  - **Swing gains independence but loses performance by not relying on native drawing calls**
  - **Has Windows, Mac, other look and feel options**
- **Supersedes Java Abstract Window Toolkit (AWT) though it still uses many non-drawing classes from that library. You will usually:**

```
import java.awt.*;
import javax.swing.*;
```

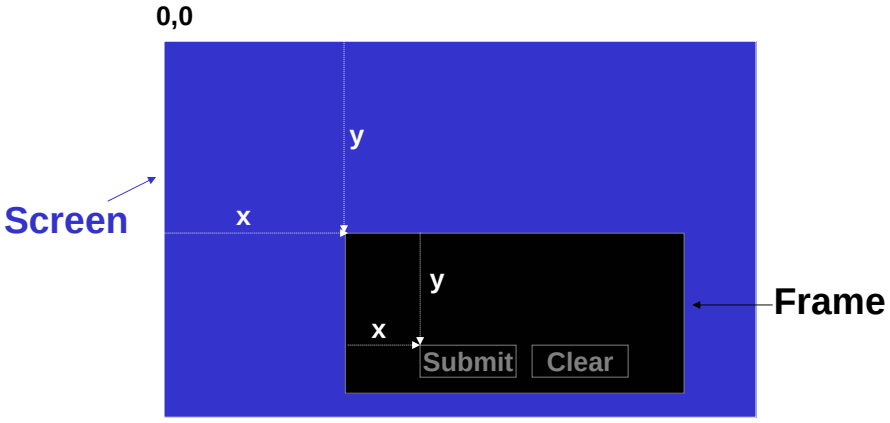
The 3 Flavors of GUI Objects

- *Top Level Windows:* are containers that are not contained by any other containers; they can be iconified or dragged and interact with the native windowing system
  - Example: `JFrame`, `JDialog`
- *Containers:* some `JComponents` are designed to hold other components, not to present info or interact with the user
  - Examples: `JPanel` , `JScrollPane`
- `JComponents`: present information or interact with the user
  - Examples: labels (`JLabel`), buttons (`JButton`), text fields (`JTextField`)
  - `JFrame` and `JDialog` are not `Jcomponents`

Anatomy of a `JFrame`



Coordinates

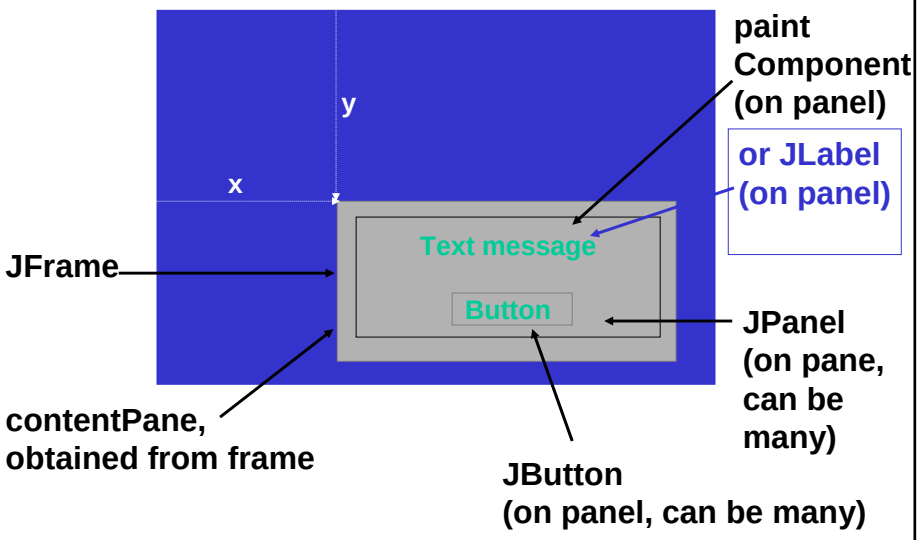


Measured in pixels (e.g. 640 by 480, 1024 by 768, etc.)
By tradition, upper left hand corner is origin (0,0)
X axis goes from left to right, y from top to bottom

Exercise 1: Empty JFrame

```
// Download, read and run this program
import javax.swing.*;

public class SwingTest {
    public static void main(String[] args) {
        // Create new frame
        JFrame frame= new JFrame();
        // Tells program to exit when user closes this frame
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         // Frame has 0 default size; give it a size
         frame.setSize(400, 300);      // setSize(int x, int y)
         // Frame invisible by default; make it visible
         frame.setVisible(true);
    }
    // main() ends but Swing "thread" stays alive
}

// Run the program; see what it draws
```

Frames, Panes and Panels



Exercise 2: Panel with Color

```
// Download, read and run this program
import java.awt.*;
import javax.swing.*;

public class SwingTest2 {
    public static void main(String args[]) {
        JFrame frame = new JFrame("Welcome to 1.00");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(500,500);
        Container contentPane= frame.getContentPane(); // No J
        // We never draw on a JFrame. Instead we update
        // components that are added to its contentPane
        JPanel panel = new JPanel();
        panel.setBackground(Color.pink);
        // Add panel to the contentPane of the JFrame
        contentPane.add(panel);
        frame.show();
    }
}
```

More Complex Drawing

- The `JFrame` in the preceding example contained a simple, colored `JPanel`
  - 13 predefined colors: `Color.x` where x is
    `orange, pink, cyan, magenta, yellow, black, blue, white, gray, lightGray, darkGray, red, green`
  - You can create your own colors (see text, section 4.5)
- If we want to accomplish more complex drawing, we need to build new UI components. These components are created by extending existing Swing classes, like `JPanel` or `JComponent` by:
  - overriding the base class `paintComponent()` method and/or
  - adding members (`JPanel`, `JButton`, etc.) and methods to the base class

Frames, Panes and Panels

- Top-level window is a Java frame (JFrame)
  - JFrame is container for user interface elements, primarily panes (contentPane)
- Programs draw UI components on panes

```
Container contentPane= getContentPane();
JButton q= new JButton("Quit");  // Buttons, text boxes, etc.
contentPane.add(q);
```

  - To draw on a pane, use inheritance:
    - Create a subclass of `JPanel` or other component to do what you want
    - Redefine (override) the `paintComponent` method in your subclass
    - `paintComponent` has `Graphics` object as argument
    - `Graphics` object stores data on fonts and colors, and has drawing methods that you can use
- Bring up class WelcomePanel in Eclipse:

Exercise 3: WelcomePanel

```
// Download WelcomePanel
import java.awt.*;
import javax.swing.*;
public class WelcomePanel extends JPanel {
    public void paintComponent(Graphics g) {
        // Have JPanel paintComponent do default operations
        // such as background color, etc.
        super.paintComponent(g);
        g.drawString("Welcome to 1.00", 125, 150);
        // The last two arguments of drawString indicate
        // that the message should be drawn at (x,y) =
        // (125,150)
    }
}
```

Exercise 3: Panel with Text

- Modify SwingTest2 main to use this new component.
  - Change:
    ```
    JPanel panel = new JPanel();
    ```
  - to:
    ```
    WelcomePanel panel = new WelcomePanel();
    ```
  - Remove:
    ```
    panel.setBackground(Color.pink);
    ```
- Run SwingTest2 main again.
  - This time we'll see a JFrame containing our new component.

Graphics

- Graphics class can draw lines, ellipses, etc.
  - Very limited: single thickness, no rotation, etc.
- Java2D library is much more functional
  - Uses Graphics2D, not Graphics objects
    - Some casting is needed (but little thought)
  - There are two versions of the Java2D library: double and float. Use the double version:
    - `Rectangle2D.Double()`
    - `Line2D.Double()`, etc.
    - (Remember these from inner classes?)
    - You can drop the `.Double` part sometimes; see examples

Exercise 4: Drawing

- To utilize the Graphics2D package, we'll create a new UI component, CirclePanel, that uses Graphics2D calls in its `paintComponent` method.
- To access the Graphics2D package, we must import the following:

```
import java.awt.geom.*;
```

- Bring up class CirclePanel in Eclipse:

Exercise 4: Drawing, p.2

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.*;     // For 2D classes
public class CirclePanel extends JPanel {
    public void paintComponent(Graphics g) {
       super.paintComponent(g);
       Graphics2D g2= (Graphics2D) g;
       int x = 200, y = 150, w = 100, h = 100;
       // Ellipse: top left corner(x,y), width = height
       Ellipse2D circle= new Ellipse2D.Double(x,y,w,h);
       g2.setPaint(Color.blue);
       g2.draw(circle);
    }
}
```

Exercise 4: Drawing, p.3

- Substitute `CirclePanel` for `WelcomePanel` in `SwingTest2` main
- When you run `SwingTest2` main, you should see a `JFrame` containing a panel that paints a blue outline of a circle.
- Add code to `CirclePanel`'s `paintComponent` method so that it:
  - Creates one `Ellipse2D`, `Rectangle2D`, and `Line2D` object.
  - Makes a different colored object
  - Makes a filled object
  - Draws some text using `Graphics2D.drawString`

Exercise 4: Drawing, p.4

- **Constructor Summary: all arguments are doubles, w is width, h is height, x,y are coordinates corresponding to upper left corner**
  - `new Ellipse2D.Double(x,y,w,h)`
  - `new Rectangle.Double(x,y,w,h)`
  - `new Line2D.Double(x1,y1,x2,y2)`
- **Code paintComponent in stages: add one 2D object at a time and check if it works as expected**
  - **To change the color that objects are painted, invoke the setPaint method on g2, your Graphics2D object. setPaint can take a single Color argument (e.g.,Color.blue)**
  - **To draw a 'filled object,' invoke the fill method on g2, your Graphics2D object. fill can take a single Ellipse2D or Rectangle2D object as its argument.**

Fonts

- **Standard constructor:**

```
Font myFont =
  new Font( String name, int style, int size );
```

- **Font name: safe approach is to use a logical font name, one of**
  - `"SansSerif", "Serif", "Monospaced", "Dialog", "DialogInput", "Symbol"`
- **Four font styles are present: Font.y where y is**
  - `Font.PLAIN, Font.BOLD, Font.ITALIC`
  - `Font.BOLD + Font.ITALIC`
- **Size is point size; 12 corresponds to standard printed text**

Exercise 5: Font

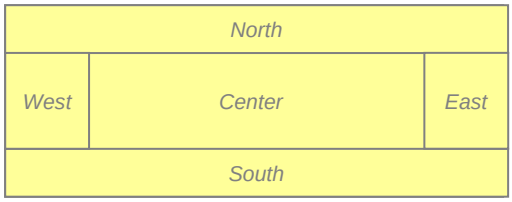
```
// Download the following program
import javax.swing.*;
import java.awt.*;
public class FontPanel extends JPanel {
    private Font[] fonts= {
      new Font("Monospaced", Font.PLAIN, 24) };

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2= (Graphics2D) g;
        int baseY= 50;
        Font f;
        for (int i= 0; i < fonts.length; i++) {
            f= fonts[i];
            g2.setFont(f);
            g2.drawString(f.getFontName() + " " +
                f.getSize(), 50, baseY);
            baseY += f.getSize() + 20;
        }
    }
}
```

Exercise 5: Font, p.2

- Substitute `FontPanel` for `CirclePanel` in `SwingTest2` main
- When you run `SwingTest2` main, you should see a `JFrame` containing a panel with the text "Monospaced.plain 24"
- Add 3 new `Font` objects to the `Font[] fonts`
  - Do this using the `Font` constructor on the previous slide with new combinations of Font names, Font styles, and Font sizes.

Building with Components

- As simple as our first application is, we can build some interesting variations with little additional code.
- `JLabel`s can hold images as well as or instead of text.
- A contentPane has 5 zones where you can add a component. Components are placed using the `BorderLayout` class



A Simple Image Viewer, 1

```
import javax.swing.*;
import java.awt.*;
import java.net.*;

public class ImageView extends JFrame {
  private URL source;
  private String title;

  public ImageView( String u, String t ) {
    title = t;
    try {
      source = new URL( u ); }
    catch ( MalformedURLException e ) {
      System.out.println( "Bad URL " + source );
      System.exit( 1 );
    }
```

A Simple Image Viewer, 2

```
   // Constructor, continued (usual approach in Swing)
    setDefaultCloseOperation( EXIT_ON_CLOSE );
    Container contentPane= getContentPane();
    // Make a label of the image from the URL
    ImageIcon image = new ImageIcon( source );
    JLabel imageLabel = new JLabel( image,
        SwingConstants.CENTER );
    contentPane.add(imageLabel, BorderLayout.CENTER );
    // Make a 2nd label of the title
    JLabel titleLabel = new JLabel( title,
        SwingConstants.CENTER );
    contentPane.add( titleLabel, BorderLayout.SOUTH );
    pack();
 }
```

A Simple Image Viewer, 3

```
public static void main( String [] args ) {
  String url =  "server";
  String title = "The New Stata Center";
  ImageView view = new ImageView( url, title );
  view.show();
 }
}
```