

Introduction to Computation and Problem Solving

Class 14: Introduction to the Swing Toolkit

Prof. Steven R. Lerman
and
Dr. V. Judson Harward

1

Class Preview

Over the next 5 lectures, we will introduce you to the techniques necessary to build graphic user interfaces for your applications.

Class 14: Introduction to Swing Basic Concepts: Components and containers, fonts, colors, borders, layout

Class 15: Constructing Swing interfaces.

Class 16: *Lab*: The Swing Event Model

Class 17: *Lab*: How to do Custom Drawing in Swing, the Graphics 2D API

Class 18: *Lab*: 2D Transformations in Swing

2

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 toolkit components
 - Has Windows, Motif, Mac look and feel options
- Supersedes Java Abstract Window Toolkit (AWT) though it still uses many non-drawing classes from that library

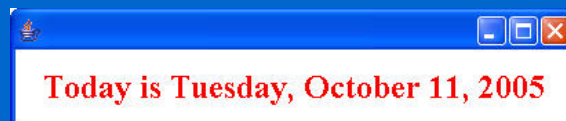
```
import java.awt.*;  
import javax.swing.*;
```

- See <http://java.sun.com/docs/books/tutorial/uiswing/index.html>

3

A Simple GUI Application

Let's build a simple application with a GUI, a date desk accessory that displays today's date:



4

Getting the Date

If we are going to display the date, we first have to figure out what it is:

```
import java.text.*;
import java.util.*;
public class Today0 {
    public static void main (String args[]) {
        DateFormat formatter =
            DateFormat.getDateInstance(DateFormat.FULL);
        Date now = new Date();
        String dateStr = formatter.format( now );
        System.out.println( dateStr );
    }
}
```

Converts Date↔Text

*Millisecond ticks since since
January 1, 1970, 00:00:00 GMT*

A readable date

5

Displaying the Date, Try 1

So, we've got the date. Let's display it. We create an instance of a GUI class containing the string date and we make it visible:

```
public static void main (String args[]) {
    DateFormat formatter =
        DateFormat.getDateInstance(DateFormat.FULL);
    Date now = new Date();
    String dateStr = formatter.format( now );
    Today1 today = new Today1( dateStr );
    today.setVisible( true );
    //today.show();
}
}
```

Our GUI class

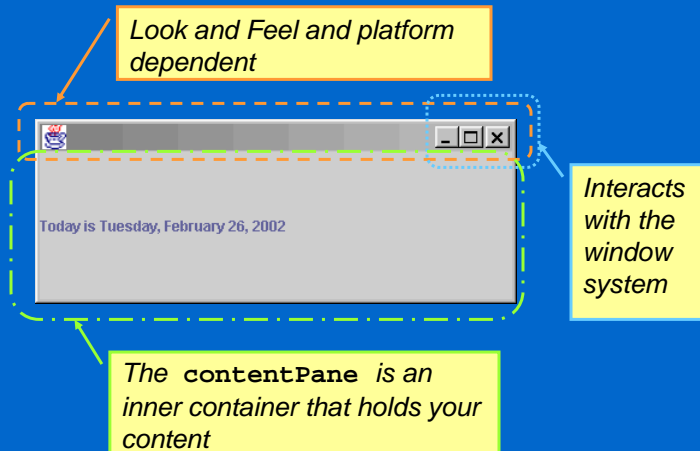
6

The 3 Flavors of GUI Classes

- **JComponents**: present information or interact with the user
 - Examples: labels (`JLabel`), buttons (`JButton`), text fields (`JTextField`)
- **Containers**: some `JComponents` are designed to hold other components, not to present or interact
 - Examples: `JPanel`, `JScrollPane`
- **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` (not `JComponents` at all)

7

Anatomy of a JFrame



8

Displaying the Date, Try 1, Part 2

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

public class Today1 extends JFrame
{
    private JLabel dateLabel;

    public Today1( String dStr ) {
        setDefaultCloseOperation( EXIT_ON_CLOSE );
        dateLabel = new JLabel( "Today is " + dStr );
        getContentPane().add( dateLabel,
                               BorderLayout.CENTER );

        pack();
    }
}
```

9

Displaying the Date, Try 1, Part 3

- The simplest way to display a string is a JLabel
- We need a JFrame to hold our JLabel

```
public class Today1 extends JFrame
{
    private JLabel dateLabel;
```

10

Displaying the Date, Try 1, Part 4

Use the ctor argument to build the JLabel

```
public Today1( String dStr ) {  
    dateLabel = new JLabel( "Today is " + dStr );  
    getContentPane().add( dateLabel,  
        BorderLayout.CENTER );  
}
```

Add the JLabel to the JFrame's contentPane

Where to add it; more on this layout manager in the next lecture

11

Displaying the Date, Try 1, Part 5

So closing the window will end the application

```
setDefaultCloseOperation( EXIT_ON_CLOSE );  
pack();  
//setSize( 300, 100 );
```

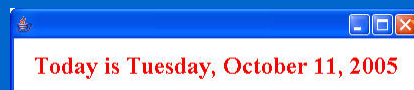
So the JFrame will fit its contents and not shrivel up like a raisin

12

Swing Application Termination

Any program that makes a Swing component visible, including a dialog created via `JOptionPane`, must explicitly exit by

1. calling `System.exit(int code)` or
2. using `setDefaultCloseOperation(EXIT_ON_CLOSE);` to tell Swing what to do when the window system attempts to close the window, (e.g. by clicking the window close box.)

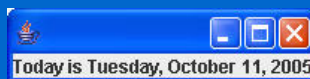


the window close box

Why? Because as soon as you put up a GUI window, you start a separate GUI *thread* that won't end when you run off the end of the `main()` method.

13

Displaying the Date, Try 1, Critique



- It's too small and the colors don't stand out.
- We need to choose a custom `Font` and text (foreground) `Color`.

```
import java.awt.Font;  
import java.awt.Color;
```

14

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**
 - PLAIN, BOLD, ITALIC
 - Font.BOLD + Font.ITALIC
- **Size is point size; 12 corresponds to standard printed text**

15

Colors

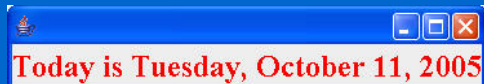
- **13 predefined colors: Color.x where x is**
 - orange, pink, cyan, magenta, yellow, black, blue, white, gray, lightGray, darkGray, red, green
- **You can define your own colors**

```
// RGB or (red-green-blue)  
Color ugly= new Color(30, 90, 120);
```

16

Displaying the Date, Try 2

```
dateFont = new Font( "Serif", Font.BOLD, 24 );
dateLabel.setFont( dateFont );
dateLabel.setForeground( Color.red );
dateLabel.setBackground( Color.white );
```



- Why doesn't the background change?
Not all `JComponents` are opaque by default.
- The `JFrame` crowds the label.

17

Borders

- In Swing, borders are objects.
- The best way to get some space around our label is to give it an empty border.
- The best way to create a border is to use the factory methods in class `BorderFactory`:

```
import javax.swing.border.*;
. . .
Border empty =
    BorderFactory.createEmptyBorder( top,left,
                                    bottom,right);
```

18

Displaying the Date, Try 3

JLabel background is transparent in some look & feels by default

```
dateLabel.setOpaque( true );  
Border empty = createEmptyBorder(10,20,10,20);  
dateLabel.setBorder( empty );  
dateLabel.setHorizontalAlignment( SwingConstants.CENTER );
```

Add our empty border to the label and center it



19

Building with Components

- As simple as our first application is, we can build some very interesting variations with little additional code.
- `JLabels` can hold images as well as or instead of text.
- A `ContentPane` has 5 zones where you can add a component.



20

A Simple Image Viewer, 1

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

public class ImageView2 extends JFrame
{
    public static void main( String [] args ) {
        String path = "gwcrcl0.jpg";
        String title = "The Leonard P. Zakim Bridge";
        ImageView2 theView = new ImageView2( path, title );
        theView.setVisible( true );
    }
}
```

21

A Simple Image Viewer, 2

```
private JLabel imageLabel;
private JLabel titleLabel;

public ImageView2( String p, String t ) {
    setDefaultCloseOperation( EXIT_ON_CLOSE );

    // make a label of an image
    ImageIcon image = new ImageIcon( p );
    imageLabel = new JLabel( image, SwingConstants.CENTER );
    getContentPane().add( imageLabel, BorderLayout.CENTER );

    // make a 2nd label of the title
    titleLabel = new JLabel( t, SwingConstants.CENTER );
    getContentPane().add( titleLabel, BorderLayout.SOUTH );
    pack();
}
```

22

Complex Components

- A `JLabel` is just about as simple as a component gets. But using a more complicated component is identical.
- Components encapsulate their functionality spatially and conceptually.
- Consider the following example that uses a `JColorChooser`, a very complicated component.

23

JColorChooser Example, 1

```
public class FavoriteColor
    extends JFrame
{
    private ColorLabel colorLabel;
    private JColorChooser chooser;

    public static void main(String args[]) {
        FavoriteColor favor = new FavoriteColor( "Jud",
            new Color( 255, 200, 100 ) );
        favor.setVisible( true );
    }
}
```

24

JColorChooser Example, 2

```
public FavoriteColor( String person, Color c ) {
    setDefaultCloseOperation( EXIT_ON_CLOSE );
    chooser = new JColorChooser( c );
    colorLabel = new ColorLabel( "This is " + person +
        "'s favorite color.", chooser );
    chooser.getSelectionModel().addChangeListener(
        colorLabel );
    getContentPane().add( chooser,
        BorderLayout.CENTER );
    getContentPane().add( colorLabel,
        BorderLayout.SOUTH );
    pack();
}
```

25