

# 1.00 Clase 14

## Introducción a la herramienta Swing

### Introducción a la clase

**Durante las 5 próximas clases, le presentaremos las técnicas necesarias para implementar interfaces gráficas de usuario para sus aplicaciones.**

**Clase 14: Introducción a los conceptos básicos de Swing: Componentes y contenedores, fuentes, colores, bordes y presentación.**

**Clase 15: Construcción de interfaces en Swing.**

**Clase 16: *Aprendizaje activo*: el modelo de eventos de Swing.**

**Clase 17: Cómo hacer diseños personalizados en Swing; *Graphics 2D API***

**Clase 18: Transformaciones 2D en Swing y estudio de caso de gráficos.**

# Swing

- El paquete de las clases de la interfaz de usuario para ventanas, menús, barras de desplazamiento, botones, etc.
- Independiente del hardware y del sistema operativo (mientras pueda diseñar una ventana):
  - Swing gana en independencia, pero pierde en rendimiento al no estar basado en las llamadas de diseño nativas.
  - Tiene como opciones de apariencia los entornos Windows, Motif y Mac
- Reemplaza a la *Abstract Window Toolkit (AWT)* de Java, aunque todavía utiliza muchas clases de esta biblioteca que no son de diseño:

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

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

## Una aplicación sencilla con una GUI

Construyamos una aplicación sencilla con una GUI, un accesorio calendario para el escritorio, que muestre la fecha actual:



## Cómo obtener la fecha

Si vamos a mostrar la fecha, primero tenemos que averiguar cuál es:

```
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 );
    }
}
```

Convierte fecha a Texto

Cuenta los milisegundos desde el 1 de enero de 1970, 00:00:00 GMT

Una fecha legible

## Mostrar la fecha, 1º intento

Ahora que tenemos la fecha, mostrémosla. Creamos una instancia de una clase GUI que contenga la fecha String y la hacemos visible:

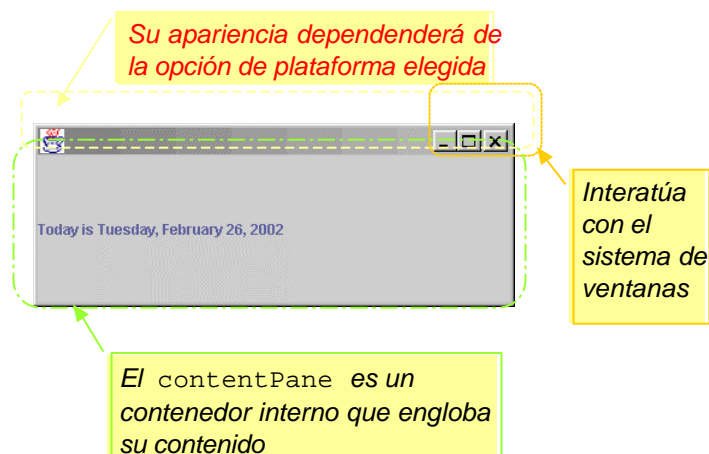
```
public static void main (String args[]) {
    DateFormat formatter =
        DateFormat.getDateInstance(DateFormat.FULL);
    Date now = new Date();
    String dateStr = formatter.format( now );
    Today1 today = new Today1( dateStr );
    today.setVisible( true );
    //today.show();
}
```

Nuestra clase GUI

## Los 3 componentes de clases GUI

- **JComponents**: presentan información o interactúan con el usuario.
  - Ejemplos: etiquetas (JLabel), botones (JButton), cuadros de texto (JTextField).
- **Containers** (contenedores): diseñados para contener a otros JComponents; no muestran información ni interactúan con el usuario.
  - Ejemplos: JPanel, JScrollPane
- **Ventanas de nivel superior**: son contenedores no sostenidos por otros contenedores cualesquiera; pueden ser iconizados o arrastrados e interactuar con el sistema de ventanas nativo.
  - Ejemplo: JFrame, JDialog (no son JComponents).

## Anatomía de un JFrame



## Mostrar la fecha, 1º intento, 2ª parte

```
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( "Hoy es " + dStr );
        getContentPane().add( dateLabel,
                               BorderLayout.CENTER );
        pack();
    }
}
```

## Mostrar la fecha, 1º intento, 3ª parte

- El modo más sencillo de mostrar una `String` es un `JLabel`.
- Nos hace falta un `JFrame` que contenga a `JLabel`.

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

## Mostrar la fecha, 1º intento, 4ª parte

Utilice el argumento ctor para construir JLabel

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

Añada JLabel al panel de contenidos de JFrame

Dónde añadirla; más sobre este gestor de distribución en la próxima clase

## Mostrar la fecha, 1º intento, 5ª parte

Al cerrar la ventana, la aplicación finalizará

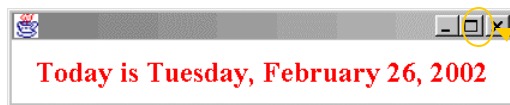
```
setDefaultCloseOperation( EXIT_ON_CLOSE );  
pack();  
//setSize( 300, 100 );
```

El tamaño de JFrame se ajustará al de sus contenidos y nunca quedará pequeño.

## Cese de una aplicación Swing

Cualquier programa que haga visible un componente Swing, inclusive un diálogo creado por medio de `JOptionPane`, debe finalizarse explícitamente:

1. llamando a `System.exit( int code )` o
2. utilizando `setDefaultCloseOperation ( EXIT_ON_CLOSE );` para indicar a Swing qué hacer cuando el sistema de ventanas trate de cerrar la ventana (ej. haciendo clic en el cuadrado de cierre de la ventana).



cuadrado de  
cierre de la ventana

¿Por qué? Porque cuando se utiliza una ventana GUI, se inicia un *hilo* GUI independiente, que no finalizará cuando se ejecute el final del método `main()`.

## Mostrar la fecha, 1º intento, crítica



- Es demasiado pequeño y los colores no resaltan.
- Es necesario elegir una Fuente (Font) personalizada y un Color de texto (primer plano).

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

## Fuentes

- **Constructor estándar:**

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

- **Nombre de la fuente:** un enfoque seguro consiste en utilizar un nombre de fuente lógico, como uno de los siguientes:
  - *"SansSerif", "Serif", "Monospaced", "Dialog", "DialogInput", "Symbol"*
- **Se presentan cuatro estilos de fuente:** *Font*.y donde y significa
  - *PLAIN, BOLD, ITALIC* (normal, negrita, cursiva)
  - *Font.BOLD + Font.ITALIC* (Fuente.NEGRITA + Fuente.CURSIVA)
- **El tamaño es el tamaño en puntos; 12 corresponde al tamaño estándar de los textos impresos.**

## Colores

- **13 colores predefinidos:** *Color.x* donde x significa:

```
- orange, pink, cyan, magenta, yellow, black,  
  blue, white, gray, lightGray, darkGray, red,  
  green
```

- **Usted puede definir sus propios colores.**

```
// RGB o (red-green-blue)(rojo-verde-azul)  
Color ugly= new Color(30, 90, 120);
```

## Mostrar la fecha, 2º intento

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



- ¿Por qué el fondo no cambia?  
No todos los `JComponents` son opacos por defecto.
- El `JFrame` se ajusta demasiado a la etiqueta.

## Bordes

- En Swing, los bordes son objetos.
- El mejor modo de ganar algún espacio alrededor de la etiqueta es asignarle un borde vacío.
- El mejor modo de crear un borde es utilizar los métodos de construcción (*factory*) de la clase `BorderFactory`:

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

## Mostrar la fecha, 3º intento

El fondo de JLabel es transparente por defecto en las opciones de apariencia

```
dateLabel.setOpaque( true );
```

```
Border empty = createEmptyBorder( 10, 20, 10, 20 );
```

```
dateLabel.setBorder( empty );
```

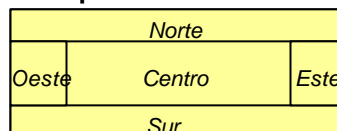
```
dateLabel.setHorizontalAlignment( SwingConstants.CENTER );
```

Añada el borde vacío a la etiqueta y céntrala.



## Implementación con componentes

- Podemos implementar algunas variaciones interesantes de una forma tan sencilla como nuestra primera aplicación, añadiendo algún código extra.
- Los componentes JLabel pueden contener imágenes además de o en vez de texto.
- Un *contentPane* (panel de contenido) posee 5 zonas en las que se puede añadir un componente.



## Un visualizador de imágenes sencillo, 1

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

public class ImageView1 extends JFrame
{
    public static void main( String [] args ) {
        String url =
        "http://web.mit.edu/buildings/statacenter/aug00home.jpg";
        String title = "The New Stata Center";
        ImageView1 theView = new ImageView1( url, title );
        theView.setVisible( true );
    }
}
```

## Un visualizador de imágenes sencillo, 2

```
private URL source;
private String title;
private JLabel imageLabel;
private JLabel titleLabel;

public ImageView1( String u, String t ) {
    setDefaultCloseOperation( EXIT_ON_CLOSE );

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

*Se explicará en una clase próxima*

## Un visualizador de imágenes sencillo, 3

```
// crear una etiqueta de una imagen
ImageIcon image = new ImageIcon( source );
imageLabel =
    new JLabel( image, SwingConstants.CENTER );
getContentPane().add(
    imageLabel, BorderLayout.CENTER );
// crear una 2ª etiqueta del título
titleLabel = new JLabel( title,
    SwingConstants.CENTER );
// ubicarlo debajo, en la zona marcada como SUR
getContentPane().add( titleLabel,
    BorderLayout.SOUTH );
pack();
}
```

## Componentes complejos

- Un `JLabel` es casi tan simple como un componente `gets`. No obstante, el uso de un componente más complicado es idéntico.
- Los componentes encapsulan su funcionalidad espacial y conceptualmente.
- Observe el siguiente ejemplo que utiliza un componente `JColorChooser`, muy complicado.

## Ejemplo 1: JColorChooser

```
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 );
    }
}
```

## Ejemplo 2: JColorChooser

```
public FavoriteColor( String person, Color c ) {
    setDefaultCloseOperation( EXIT_ON_CLOSE );
    chooser = new JColorChooser( c );
    colorLabel = new ColorLabel( "Este es el color favorito de"
        + person + ".", chooser );
    chooser.setPreviewPanel( colorLabel );
    chooser.getSelectionModel().addChangeListener(
        colorLabel );
    getContentPane().add( chooser,
        BorderLayout.CENTER );
    pack();
}
```