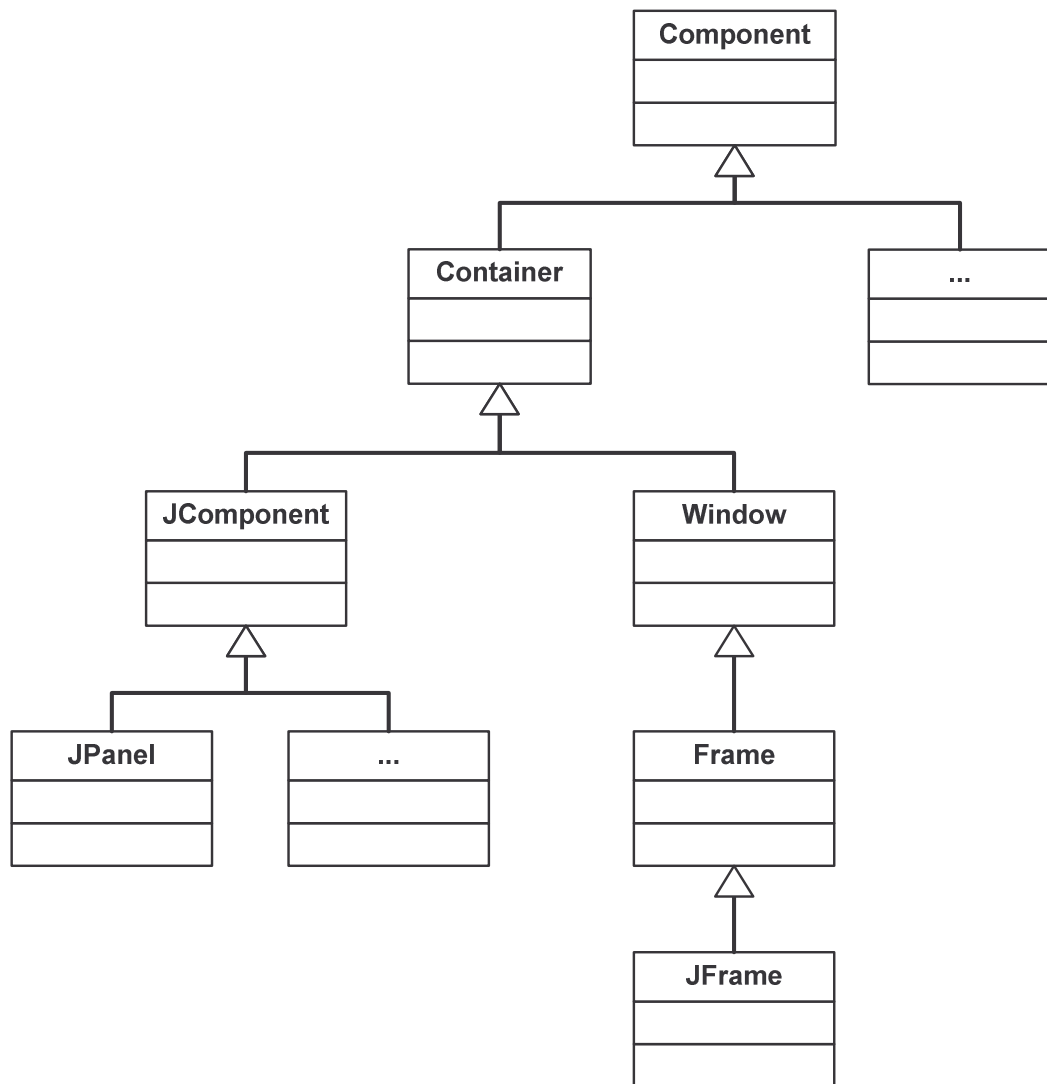


Componentes Swing
















Las clases cuyo nombre comienza por **J** forman parte de Swing. Todas las demás están incluidas en AWT (Abstract Window Toolkit)

- Component es una clase abstracta que representa a cualquier componente con representación gráfica.
- Container es un componente que puede contener a otros componentes gráficos (p.ej. JPanel).
- JFrame permite representar ventanas, si bien también se pueden utilizar clases como JApplet o JDialog.

Componentes estándar

Los **frames** (como `JFrame`) son contenedores, por lo que incluyen un “panel de contenido” (*content pane*) al cual se le pueden añadir componentes gráficos (etiquetas, botones, cajas de texto, etc.) y otros contenedores (como paneles `JPanel`).

Las interfaces gráficas de usuario se construyen con componentes, cada uno de los cuales está preparado para responder a distintos tipos de eventos. Algunos de los componentes incluidos en Swing son:

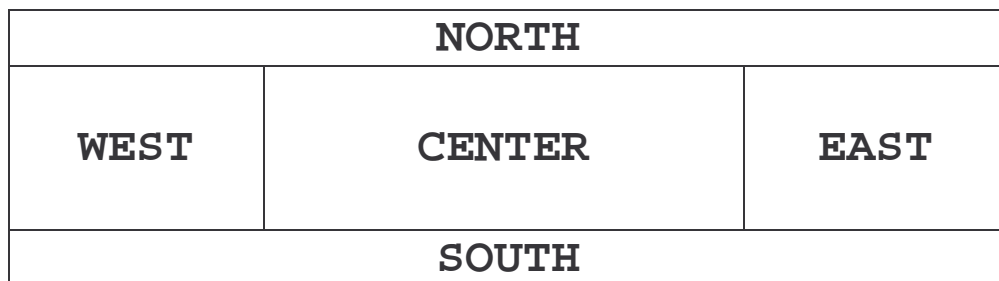
-  `JLabel` (etiqueta para mostrar texto)
-  `JTextBox` & `JTextArea` (cajas de texto para la entrada de datos)
-  `JButton` (botón)
-  `JCheckBox` (caja de comprobación, para elegir opciones)
-  `JRadioButton` (para elegir opciones mutuamente excluyentes)
-  `JList` (lista de opciones)
-  `JComboBox` (lista desplegable de opciones)
-  `JScrollBar` (barra de scroll)
-  `JTree` (árbol)
-  `JTable` (tabla)
-  Menús: `JMenuBar`, `JMenu`, `JMenuItem`
-  Ventanas de diálogo: `JOptionPane`
-  Ventanas estándar: `JFileChooser`, `JColorChooser`

Distribución del espacio (layout management)

Antes de construir una interfaz gráfica de usuario es importante saber cómo se distribuyen especialmente los componentes.

Los *layout managers* controlan la forma en la que colocan los componentes dentro de un contenedor:

- `FlowLayout` (por defecto para un `JPanel`) coloca los componentes de izquierda a derecha, de arriba abajo. Los componentes se van colocando a la derecha de los ya existentes hasta que se añade una nueva fila cuando no queda espacio suficiente en la fila actual. Cuando el contenedor se redimensiona, los componentes se redistribuyen automáticamente.
- `BorderLayout` (por defecto para un `JFrame`) permite dividir el espacio disponible de la siguiente forma:



Usualmente,
al `JFrame` se le añaden paneles usando `BorderLayout`
y luego se rellenan los paneles `JPanel` usando `FlowLayout`

Existen otros layout managers más sofisticados (como `GridLayout` o `GridBagLayout`) e incluso podemos crear nuestros propios gestores de layout si implementamos las interfaces `java.awt.LayoutManager` y `java.awt.LayoutManager2`.

Dibujo de gráficos 2D

Los **paneles** (como `JFrame`) son contenedores que pueden contener otros componentes y, además, tienen una superficie sobre la que se puede dibujar (*canvas* o lienzo).

Para dibujar en un panel `JPanel`, hay que redefinir el método `paintComponent()` de la clase `JComponent`, que se invoca automáticamente cada vez que hay que refrescar la visualización del componente en la pantalla (p.ej. cuando se mueve o redimensiona una ventana).

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

class JPanelFrame extends JFrame
{
    public JPanelFrame()
    {
        this.setTitle("Mi primer programa gráfico");
        this.setSize(400,150);
        this.addWindowListener(new MainWindowListener());
        this.getContentPane().add(new MiPanel());
    }
}

class MainWindowListener extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
}

public class JPanelTest
{
    public static void main(String[] args)
    {
        JFrame frame = new JPanelFrame();
        frame.setVisible(true);
    }
}
```

```

class MiPanel extends JPanel
{
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);

        // Figuras geométricas
        g.setColor(Color.RED);
        g.drawRect(20,20,360,60);

        g.setColor(Color.CYAN);
        int[] xcoords={25,35,365,375,375,25};
        int[] ycoords={35,25, 25, 35, 75,75};
        g.drawPolygon(xcoords,ycoords,6);
        g.fillPolygon(xcoords,ycoords,6);

        g.setColor(Color.LIGHT_GRAY);
        g.drawOval(100,30,90,60);
        g.fillOval(100,30,90,60);

        // Mensaje de texto
        Font f = new Font("Helvetica",Font.BOLD,25);
        g.setFont(f);
        g.setColor(Color.WHITE);
        g.drawString("¡Hola!",250,60);
        g.setColor(Color.BLACK);
        g.drawString("¡Hola!",248,58);
    }
}

```

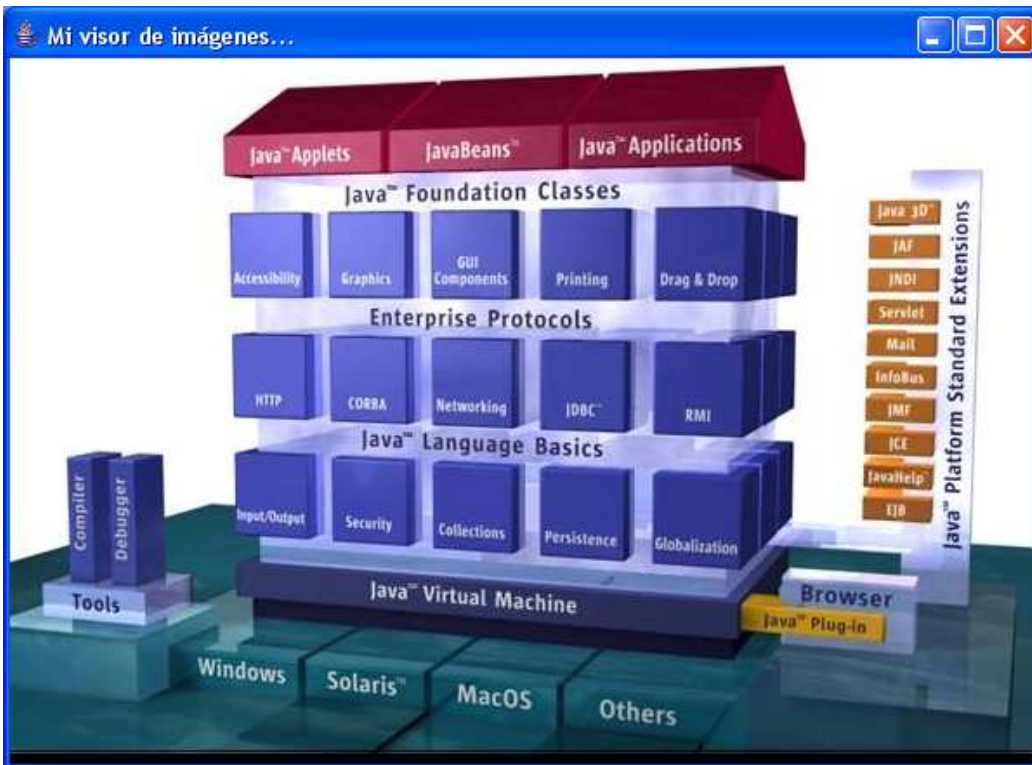


Un objeto de tipo `java.awt.Graphics` define el contexto en el que se dibuja (tipo de letra, estilo de línea, color...), e incluye métodos para dibujar distintos tipos de figuras y mostrar mensajes de texto.

Visualización de imágenes

El método `Graphics.drawImage()` nos permite dibujar imágenes en formato GIF, PNF o JPEG.

Las imágenes podemos leerlas con el método `getImage()` de la clase `java.awt.Toolkit`



```
class ImagePanel extends JPanel
{
    Image img;

    public ImagePanel ()
    {
        img=Toolkit.getDefaultToolkit().getImage("java.jpg");
    }

    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        if (img!=null)
            g.drawImage(img,0,0,this);
    }
}
```

El resto del código es igual que en los ejemplos anteriores:

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

class ImageFrame extends JFrame
{
    public ImageFrame()
    {
        setTitle("Mi visor de imágenes...");
        setSize(600,440);
        addWindowListener(new MainWindowListener());

        Container contenido = getContentPane();
        contenido.add(new ImagePanel());
    }
}

class MainWindowListener extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
}

public class ImageTest
{
    public static void main(String[] args)
    {
        JFrame frame = new ImageFrame();
        frame.setVisible(true);
    }
}
```