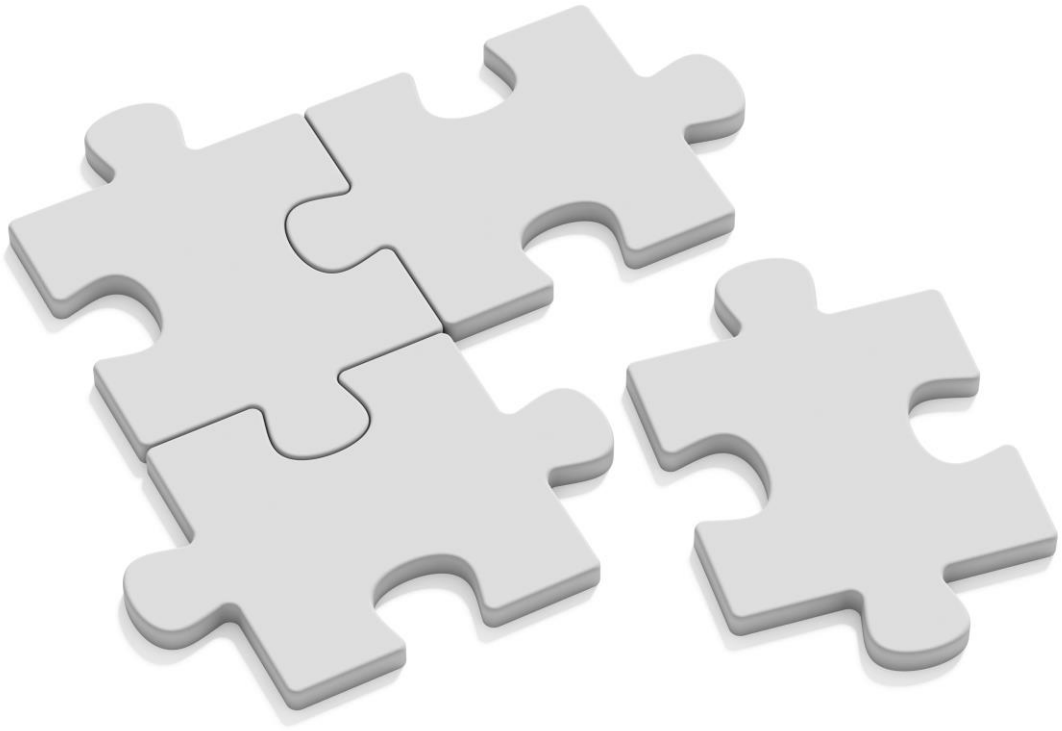
Universidad Tecnológica Nacional



2017

CÁTEDRA DE LENGUAJE DE PROGRAMACIÓN JAVA

Ings. Mario Bressano & Miguel Iwanow

ENVÍO 05/2017

¿Qué es un Applet?

Es otra manera de incluir código a ejecutar en los clientes que visualizan una página web. Se trata de pequeños programas hechos en Java, que se transfieren con las páginas web y que el navegador ejecuta en el espacio de la página.

Los applets de Java están programados en Java y precompilados, es por ello que la manera de trabajar de éstos varía un poco con respecto a los lenguajes de script como Javascript. Los applets son más difíciles de programar que los scripts en Javascript y requerirán unos conocimientos básicos o medios del lenguaje Java.

La principal ventaja de utilizar applets consiste en que son mucho menos dependientes del navegador que los scripts en Javascript, incluso independientes del sistema operativo del ordenador donde se ejecutan. Además, Java es más potente que Javascript, por lo que el número de aplicaciones de los applets podrá ser mayor.

Como desventajas en relación con Javascript cabe señalar que los applets son más lentos de procesar y que tienen espacio muy delimitado en la página donde se ejecutan, es decir, no se mezclan con todos los componentes de la página ni tienen acceso a ellos. Es por ello que con los applets de Java no podremos hacer directamente cosas como abrir ventanas secundarias, controlar Frames, formularios, capas, etc.

Ejemplo de código de Applet:

```
<HTML>
<BODY>
  <APPLET CODE="Hola.class" WIDTH="200" HEIGHT="70">
  </APPLET>
</BODY>
</HTML>

/**
 * Applet AdiosMundo
 *
 * <APPLET CODE="Hola.class" WIDTH="200" HEIGHT="70"></APPLET>
 */

import java.applet.Applet;
import java.awt.*;

public class AdiosMundo extends Applet {
  public void paint(Graphics g) {
    g.drawString("Java Applet",20,20);
  }
```


```
}
```

Vamos ahora a examinar el código paso a paso:

```
import java.applet.Applet;
import java.awt.*;
```

En todos nuestro applets deberemos, como mínimo, incluir estas dos sentencias de importación. En la primera importamos la clase Applet, de la cual derivaremos nuestros applets. La segunda permite importar las clases de la librería AWT, que nos permitirán dibujar en el rectángulo asignado al applet.

```
public class Hola extends Applet {
}
```

Nuestros applet siempre van a ser clases derivadas de Applet, pues en esta clase y en sus clases padre se definen los métodos que deberemos sobreescribir para que nuestros applets respondan a los eventos generados desde el navegador.

```
 public void paint(Graphics g) {
 }
```

Este es el método que debemos sobreescribir para dibujar en el rectángulo asignado a nuestro applet. Recibe un parámetro que es una referencia a una instancia de tipo Graphics. Esta clase permite definir lo que se suele denominar lienzo (o Canvas). El lienzo es precisamente ese rectángulo del que estamos hablando todo el rato.

```
   g.drawString("Hola",20,20);
```

La clase Graphics dispone de muchos métodos para dibujar en el lienzo. Éste, concretamente, nos permite dibujar un texto en la posición que queramos. En este caso el texto se pintará 20 pixels a la derecha del borde izquierdo de la pantalla y otros 20 más abajo del borde de arriba.

Ciclo de vida de un applet

Como hemos visto, un applet se ejecuta en una página web y ese entorno en el que está va a determinar el ciclo de vida del mismo.

La clase Applet dispone de varios métodos sin código dentro a los que llama en diversas fases de su ciclo de vida (al crearse el applet, al redibujarse, al destruirse...). Si deseamos que nuestro applet haga cosas especiales en esos momentos deberemos sobreescribir esos métodos. Son éstos:

void init()
: Este método se llama al crearse el applet y sólo se ejecuta esa vez. Se suele utilizar para inicializar variables globales.

void start()

Se llama cuando hay que activar el applet. Esto sucede cuando el applet se ve en la pantalla. Se suele utilizar para comenzar las actividades de applet, ya que en general éste debe estar visible en la pantalla para poder realizar su función.

void stop()

Es el inverso del anterior. Es llamado cuando el applet deja de estar visible. Se suele utilizar para interrumpir las actividades activadas en start().

void destroy()

Método llamado al destruirse el applet. Normalmente no se suele sobreescribir, ya que la destrucción del applet conlleva la destrucción de todo lo que se ha creado en él. Puede ser útil para mostrar una ventana de confirmación que pregunte al usuario si de verdad debe cargárselo.

void repaint()

Este método no debería ser sobrecargado y es llamado cuando se necesita redibujar el applet. Podemos llamarlo nosotros desde nuestro código. Llama a update(Graphics).

void update(Graphics)

Llamado por repaint() cuando se necesita actualizar el applet. Su implementación borra el rectángulo y llama a paint() para volver a pintarlo.

void paint(Graphics)

Método en el que se dibuja el applet y el cual deberemos sobreescribir para poder dibujar lo que nosotros deseemos..

void print(Graphics)

Equivalente a paint() pero es llamado cuando se va a imprimir el contenido del applet.

El siguiente ejemplo conviene observarlo desde un navegador e ir viendo los distintos mensajes que nos muestra el applet, para comprender cuando son llamados los métodos:

CicloDeVida.java

```
/**
 * Applet CicloDeVida
 *
 * <APPLET CODE="CicloDeVida.class" WIDTH="100" HEIGHT="400"></APPLET>
 */

import java.applet.Applet;
import java.awt.Graphics;

public class CicloDeVida extends Applet {
  String mensajes;
  public void init() {
    mensajes = "Inicializando. ";
  }
  public void start() {
    escribirMensaje("Método start. ");
  }
  public void stop() {
    escribirMensaje("Método stop. ");
  }
  public void destroy() {
    escribirMensaje("Destruyendo Applet. ");
  }
  void escribirMensaje(String nuevoMensaje) {
    mensajes += nuevoMensaje;
```


```
    repaint();
  }
  public void paint(Graphics g) {
    g.drawString(mensajes, 5, 15);
  }
}
```

Etiqueta APPLET

La etiqueta APPLET presenta varios parámetros, de los cuales sólo es obligatorio poner los ya comentados CODE, WIDTH y HEIGHT. Son los siguientes:

CODE
: Nombre completo (incluyendo extensión) del fichero que contiene el applet.

WIDTH
: Anchura del rectángulo donde se ejecutará el applet.

HEIGHT
: Altura del rectángulo donde se ejecutará el applet.

CODEBASE
: Dirección donde está el fichero .class que contiene al applet. Es necesario ponerlo cuando el applet se encuentra en un directorio distinto al de la página desde la que se le llama, ya que CODE no puede contener directorios, sólo el nombre del fichero.

ALT
: Algunos navegadores comprenden la etiqueta APPLET pero no pueden mostrar applets. Esto es debido a no tener instalada la máquina virtual Java o a que son navegadores en modo texto. En ese caso mostrarán el contenido de este parámetro en lugar del applet.

Hay alguno más, pero de poca importancia.

Paso de parámetros

Entre <APPLET> y </APPLET> podremos colocar etiquetas PARAM que nos permitirán pasar parámetros al applet. Tienen dos atributos:

VALUE
: Nombre del parámetro.

NAME
: Valor del parámetro.

Podemos obtener esos valores por medio del método getParameter(String), como vemos en el siguiente ejemplo:

MostrarMensaje.java

```
/**
 * Applet MostrarMensaje
 *
 * <APPLET CODE="MostrarMensaje.class" WIDTH="200" HEIGHT="70">
 *   <PARAM NAME="Mensaje" VALUE="Mi mensaje propio">
 * </APPLET>
 */

import java.applet.Applet;
```


```
import java.awt.*;

public class MostrarMensaje extends Applet {
  String mensaje;
  public void init() {
    mensaje = getParameter("Mensaje");
  }
  public void paint(Graphics g) {
    g.drawString(mensaje,20,20);
  }
}
```

Hay que destacar que Java no distingue entre mayúsculas y minúsculas en cuanto al nombre de los parámetros.

En muchos casos, el usuario puede que no incluya parámetros que consideramos necesarios o que escriba mal el nombre de un parámetro. En ese caso, la llamada a getParameter() nos devolverá null. Debemos tener cuidado con esto, ya que nos pueden saltar excepciones por esta clase de cosas. Así pues, el código correcto de init() será:

```
public void init() {
  mensaje = getParameter("Mensaje");
  if (mensaje==null)
    mensaje = "Mensaje por defecto";
}
```

De este modo, si nos equivocamos en la etiqueta PARAM, nos mostrará un mensaje y no lanzará ninguna excepción.

Clase Graphics

La clase Graphics dispone de más métodos aparte de drawString, que nos permitirán dibujar figuras e imágenes.

Métodos de dibujo

Mientras no se especifique, consideraremos que todos los parámetros son enteros:

drawString(String texto,x,y)
Escribe un texto a partir de las coordenadas (x,y).
drawLine(x1,y1,x2,y2)
Dibuja una línea entre las coordenadas (x1,y1) y (x2,y2).
drawRect(x,y,ancho,alto)
fillRect(x,y,ancho,alto)
clearRect(x,y,ancho.alto)
Son tres métodos encargados de dibujar, rellenar y limpiar, respectivamente, un rectángulo cuya esquina superior izquierda está en las coordenadas (x,y) y tienen el ancho y alto especificados.
drawRoundRect(x,y,ancho,alto,anchoArco,altoArco)
fillRoundRect(x,y,ancho,alto,anchoArco,altoArco)

Equivalentes a los anteriores, pero con las esquinas redondeadas. La forma y tamaño de dichas esquinas viene dada por los dos últimos parámetros.

draw3DRect(x,y,ancho,alto,boolean elevado)
fill3DRect(x,y,ancho,alto,boolean elevado)

Equivalentes a los primeros, pero dibujan un borde para dar la sensación de que está elevado o hundido (dependiendo del valor del último parámetro).

drawOval(x,y,ancho,alto)
fillOval(x,y,ancho,alto)

Dibujan una elipse con esquina izquierda superior en (x,y) y el ancho y alto especificados. Si son iguales dibujará un círculo.

drawArc(x,y,ancho,alto,anguloInicio,anguloArco)
fillArc(x,y,ancho,alto,anguloInicio,anguloArco)

Dibuja una arco cuyo primer vértice está en (x,y) y el segundo en (x+ancho,y+alto). La forma del mismo vendrá dado por los dos últimos parámetros.

drawPolygon(int[] coordenadasX,int[] coordenadasY,numCoordenadas)
fillPolygon(int[] coordenadasX,int[] coordenadasY,numCoordenadas)

Dibuja un polígono cerrado del número de puntos especificado en el último parámetro.

copyArea(xOrigen,yOrigen,ancho,alto,xDest,yDest)

Copia el área cuya esquina superior izquierda está en (xOrigen,yOrigen) y de ancho y alto especificados a la zona que comienza en (xDest, yDest).

Por ejemplo, podemos dibujar lo siguiente:

Figurines.java

```
/**
 * Applet Figurines
 *
 * <APPLET CODE="Figurines.class" WIDTH="200" HEIGHT="70"></APPLET>
 */

import java.applet.Applet;
import java.awt.*;

public class Figurines extends Applet {
  public void paint(Graphics g) {
    g.drawString("Adios, mundo cruel",20,20);
    g.fillRoundRect(10,30,50,20,10,10);
    g.draw3DRect(100,30,50,20,true);
  }
}
```

Clase Color

Ahora ya sabemos como dibujar, sin embargo todo nos sale en un aburrido blanco y negro... Graphics almacena internamente el color con el que pinta todo, que por defecto es negro. Para poder cambiar dicho color deberemos utilizar el método setColor(), que recibe un único parámetro de tipo Color.

La clase Color dispone de varias propiedades estáticas que contienen los colores más comunes; son Color.black (negro), Color.blue (añil), Color.cyan (azul), Color.darkGray (gris claro), Color.gray (gris), Color.green (verde), Color.lightGray (verde claro), Color.magenta (magenta),

Color.orange (naranja), Color.pink (rosa), Color.red (rojo), Color.white (blanco) y Color.yellow (amarillo). Pero también se pueden crear instancias de color por medio de constructores para definir colores menos comunes. Por ejemplo:

Color c = new Color(int rojo, int verde, int azul)
: El color creado tendrá las cantidades de rojo, verde y azul indicadas, las cuales deberán estar en un rango que va de 0 a 255.

Color c = new Color(int rojo, int verde, int azul, int alfa)
: Idéntico al anterior, pero añade un canal alfa que indica el grado de transparencia. Sólo funciona con el JDK 1.2.

Cambiando el ejemplo anterior:

FigurinesColor.java

```
/**
 * Applet FigurinesColor
 *
 * <APPLET CODE="FigurinesColor.class" WIDTH="200" HEIGHT="70"></APPLET>
 */

import java.applet.Applet;
import java.awt.*;

public class FigurinesColor extends Applet {
  public void paint(Graphics g) {
    g.drawString("Adios, mundo cruel",20,20);
    g.setColor(Color.green);
    g.fillRoundRect(10,30,50,20,10,10);
    g.setColor(new Color(30,40,50,100));
    g.fill3DRect(50,40,50,20,true);
  }
}
```

Manejo de imágenes

Aún cuando podemos hacer cosas chulas con figuras, no estaría de más poder hacer cosas con imágenes, que son bastante más espectaculares. Para hacerlo deberemos utilizaremos el siguiente método:

drawImage(Image img,int x,int y,ImageObserver observador)

Vemos que necesitamos un objeto Image y otro ImageObserver y de esas cosas no sabemos todavía... No vamos a ver estos objetos muy a fondo. ImageObserver es un interfaz (que un antepasado de Applet implementa) que nos ofrece información sobre la imagen según la imagen se carga de la red. Por otro lado, Image es un objeto que representa a la imagen. Normalmente crearemos un objeto Image por medio del método de Applet getImage():

Image img = getImage(URL base, String fichero)
: Asigna la imagen contenida en la URL donde se encuentre el fichero que contiene la imagen que queremos presentar y el nombre de ese fichero.

Normalmente obtendremos esa dirección a partir de la página donde esté la página HTML o nuestro applet. Para averiguarlo disponemos de dos métodos en la clase Applet:

URL getDocumentBase()
Devuelve la URL donde está la página HTML (excluyendo el nombre del fichero).
URL getCodeBase()
Devuelve la URL donde está el applet (excluyendo el nombre del fichero).

Si deseamos obtener una URL distinta deberemos importar esa clase (java.net.URL) y crear uno pasándole una cadena como parámetro al constructor conteniendo la URL.

Normalmente, se realiza el getImage() en el método init() del applet, ya que en el momento en que ejecutemos este método comenzará la carga de imágenes desde Internet y eso conviene hacerlo cuanto antes. Para mostrar la imagen se utiliza el método paint(), al igual que hemos hecho para pintar cosas. Por ejemplo, suponiendo que tengamos un fichero llamado home.jpg:

Grafico.java
```
/**
 * Applet Grafico
 *
 * <APPLET CODE="Grafico.class" WIDTH="200" HEIGHT="70"></APPLET>
 */

import java.applet.Applet;
import java.awt.*;

public class Grafico extends Applet {
  Image img;
  public void paint(Graphics g) {
    img = getImage(getDocumentBase(), "home.jpg");
  }
  public void paint(Graphics g) {
    g.drawImage(img,x,y,this);
  }
}
```

EJERCICIO 42: Sea un applet en el cual ingresado un número se muestre el mismo al presionar un botón.

```
//Clase en donde se encuentra el applet
import java.awt.*;

//Clase en donde se encuentra evento del botón
import java.awt.event.ActionEvent;

//Clase en donde se encuentra el evento de escucha de la acción sobre el botón
import java.awt.event.ActionListener;
import javax.swing.*;
```


```
//Definimos la clase y le heredamos las características del Applet, implementamos la Acción de escucha
public class EstoEsUnApplet extends JApplet implements ActionListener {

        JTextArea areaS = new JTextArea();

        //Inicializo las variables
        JLabel etiquetaDelTitulo;
        JTextField campoIngresoDato,campoResultado;
        JButton boton;

        public void init()
        {
                //Definino clase Contenedor

Container contenedor= getContentPane();

                //Definino ubicación en el Applet con FlowLayout

contenedor.setLayout(new FlowLayout());

                etiquetaDelTitulo  =new JLabel("Ingresa un teléfono: ");
                contenedor.add(etiquetaDelTitulo);
                campoIngresoDato  =new JTextField (20);
                contenedor.add(campoIngresoDato);
                campoResultado  =new JTextField (20);
                campoResultado.setEditable(false);
                contenedor.add(campoResultado);

                boton  =new JButton("Ejecutar");
                contenedor.add(boton);
                //Registrar applet para que se escuchen las acciones

                boton.addActionListener(this);

                //Fin del método INIT
        }

        public void actionPerformed (ActionEvent actionEvent)
        {
                int telefono;
                String resultado;
                telefono= Integer.parseInt(campoIngresoDato.getText());
                resultado="El número de teléfono es "+telefono;
                campoResultado.setText(resultado);



        }//Fin de Action Performed
}
```

Ejercicio 42: Ingresado un valor calcular el factorial. Utilizar Applet y Métodos. (ActionListener)

Ejercicio 43: Convertir en un Applet al ejercicio que permitía crear una aplicación en la cual se ingresen dos ángulos entre 0 y 90º y nos de el valor en radianes. Utilizar métodos y las sentencias conocidas para ingreso y salida. Analizar los errores observados.

Ejercicio 44: Se pide realizar una aplicación en Applet que permita calcular utilizando métodos el sueldo de los operarios. Se conoce que hay 3 categorías (Cat. 1: 10,00$/hora, Cat. 2: 15,00$/hora y Cat. 3: 20,00$/hora). Por otro lado su el empleado trabaja más de 40 hs por semana percibe sobre las horas excedentes una bonificación del 25% sobre el sueldo por cada hora. Son datos el nombre del obrero, cantidad de horas trabajadas y su categoría.

Ejercicio 45: Se pide realizar una aplicación en Applet que permita calcular el tiempo de permanencia de un vehículo y el valor de la estadía. Los datos a ingresar son Hora de entrada, minutos de entrada, hora de salida, minutos de salida, categoría del vehículo.
Los valores son:
Cat.1 = auto chico, 1$ la media hora, luego 0.35$ por cada 15 minutos
Cat.2 = auto mediano, 1,5$ la media hora, luego 0.45$ por cada 15 minutos
Cat.1 = camioneta, 2$ la media hora, luego 0.55$ por cada 15 minutos
La playa abre a las 7.00 hs y cierra a las 24.00 hs

Ejercicio 46: Se debe realizar una aplicación Applet que permita calcular ingresados el nombre del alumno y 5 notas (valor real), el valor del promedio (valor entero, se redondea: 3.50=4, 3,49=3) y su condición.
Las condiciones son:
Promedio>=7……Aprobado
Promedio<4……Marzo
Promedio entre 4 y 7……Diciembre

Ejercicio 47: Crear una aplicación en la cual se ingresen dos ángulos entre 0 y 90º y nos de el valor en radianes. Utilizar métodos y las sentencias conocidas para ingreso y salida. Analizar los errores observados.

Ejercicio 48: Ingresado un ángulo (entre 0 y 90) en grados, minutos y segundos, calcular las funciones trigonométricas del mismo. Utilizar métodos y las sentencias conocidas para ingreso y salida. Analizar los errores observados.

RESOLUCIÓN EJERCICIO 48

Ejercicio 48: Ingresado un ángulo (entre 0 y 90) en grados, minutos y segundos, calcular las funciones trigonométricas del mismo


```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.*;

public class Trigonom extends JApplet implements ActionListener
{
	JTextArea areaS = new JTextArea();

	//Inicializo las variables
	JLabel etiquetaTitulo1, etiquetaTitulo2, etiquetaGrado, etiquetaMinuto,
etiquetaSegundo , etiquetaSeno, etiquetaCoseno, etiquetaTangente;
	JTextField campoGrado, campoMinuto, campoSegundo, campoSeno, campoCoseno,
campoTangente;
	JButton boton;

	public void init()
	{
		Container contenedor= getContentPane();

		contenedor.setLayout(new FlowLayout());

		etiquetaTitulo1  =new JLabel("Cálculo de Funciones Trigonomètricas de un
ángulo");
		contenedor.add(etiquetaTitulo1);
		etiquetaTitulo2  =new JLabel("Ingrese un ángulo..");
		contenedor.add(etiquetaTitulo2);


		etiquetaGrado  =new JLabel("Grados  :");
		contenedor.add(etiquetaGrado);
		campoGrado  =new JTextField (4);
		contenedor.add(campoGrado);
		etiquetaMinuto  =new JLabel("Minutos :");
		contenedor.add(etiquetaMinuto);
		campoMinuto  =new JTextField (4);
		contenedor.add(campoMinuto);
		etiquetaSegundo  =new JLabel("Segundos:");
		contenedor.add(etiquetaSegundo);
		campoSegundo  =new JTextField (4);
		contenedor.add(campoSegundo);

		boton  =new JButton("Calcular");
		contenedor.add(boton);
		//Registrar applet para que se escuchen las acciones

		boton.addActionListener(this);


		campoSeno  =new JTextField (55);
		campoSeno.setEditable(false);
```


```
                contenedor.add(campoSeno);
                campoCoseno  =new JTextField (55);
                campoCoseno.setEditable(false);
                contenedor.add(campoCoseno);
                campoTangente  =new JTextField (55);
                campoTangente.setEditable(false);
                contenedor.add(campoTangente);
        //Fin del método INIT
        }
        public void actionPerformed (ActionEvent actionEvent)
        {


        double gra, min, seg, tot, totRad;
        String seno, coseno, tangente;
        gra= Integer.parseInt(campoGrado.getText());
        min= Integer.parseInt(campoMinuto.getText());
        seg= Integer.parseInt(campoSegundo.getText());
        if (gra<0||gra>90)
                {
                        showStatus ("El valor del ángulo en grados no es permitido");
                        JOptionPane.showMessageDialog(null, "Valor de grados no
permitido= "+gra, "Error",JOptionPane.ERROR_MESSAGE);
                        System.exit(0);
                }
          else
                 if (min<0||min>59)
                        {
                                showStatus ("El valor de los minutos del ángulo no es
permitido");
                                JOptionPane.showMessageDialog(null, "Valor de
minutos no permitido= "+min,"Error",JOptionPane.ERROR_MESSAGE);
                                System.exit(0);
                        }
              else
                                if (seg<0||seg>59)
                        {
                                showStatus ("El valor de los segundos del ángulo no
es permitido");
                                JOptionPane.showMessageDialog(null, "Valor de
segundos no permitido= "+seg,"Error",JOptionPane.ERROR_MESSAGE);
                                System.exit(0);
                        }
        showStatus ("Calculando…");

        tot = gra + (min/60)+ (seg/3600);
        totRad=tot*Math.PI/180;
        seno="El valor del seno es "+Math.sin(totRad);
        coseno="El valor del coseno es "+Math.cos(totRad);
        tangente="El valor de la tangente es "+Math.tan(totRad);
        showStatus ("Terminado");
        campoSeno.setText(seno);
        campoCoseno.setText(coseno);
```


```
        campoTangente.setText(tangente);
        }//Fin de Action Performed

}
```

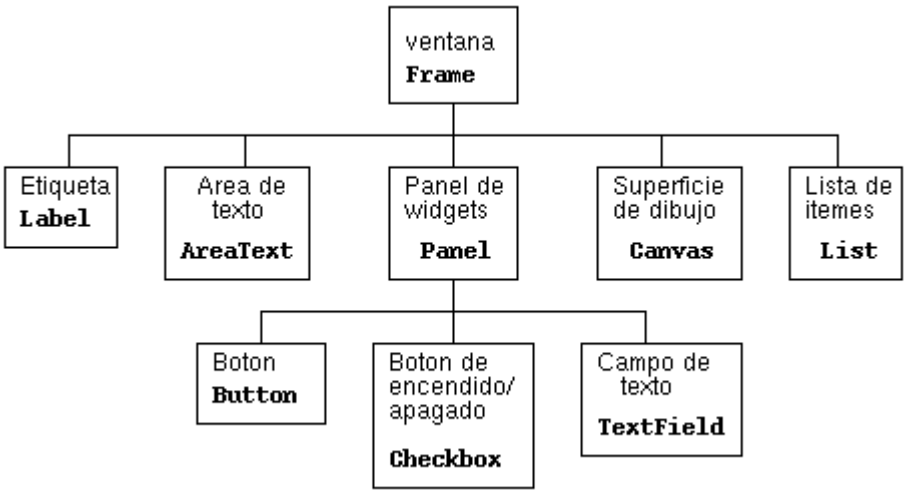
Hablando un poco más en detalle del Kit Gráfico

El Kit Gráfico: AWT

- AWT: *Abstract Window Toolkit*.
- Un *widget* es un elemento gráfico con el que el usuario puede interactuar.
- Ocupa una porción rectangular en una ventana.
- Existen diversos tipos: botones, áreas de texto, etiquetas, etc.
- El kit AWT implementa los widgets más usuales. Para usarlos es necesario importar las clases del paquete java.awt:
    - import java.awt.*;

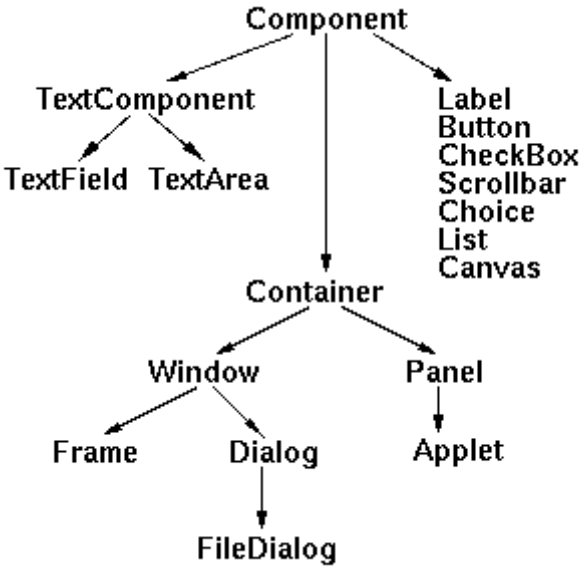
Algunos Widgets de la AWT

Los widgets se organizan en una jerarquía cuya raíz es la ventana de la aplicación.



- Los botones, áreas de texto, etiquetas, etc. son hojas en la jerarquía.
- Los nodos internos se denominan *contenedores*:
    - Frame: Una ventana.
    - Panel: Una porción rectangular de una ventana pre-existente.
    - Applet: Un panel dentro de un browser Web.
    - FileDialog: Una ventana para que el usuario seleccione un archivo.
- Los widgets se implementan como extensiones de la clase Component.

Jerarquía de clases de AWT



Construcción de una ventana de interacción

- Construcción del Frame:
  - Frame frame= new Frame("Titulo de la ventana");
- Construcción de widgets básicos de interacción:

Botón:	**Button boton= new Button("nombre del boton");**
Etiqueta:	**Label etiq= new Label("nombre", Label.CENTER);**
Área de texto:	**TextArea area= new TextArea(5, 20);**
Campo de texto:	**TextField texto= new TextField(15);**
Botón on/off:	**Checkbox check= new Checkbox("nombre");**

La lista de ítemes:

```
List lista = new List(3);
for (int i = 1; i <= 10; i++) {
  lista.addItem("List item " + i);
}
```

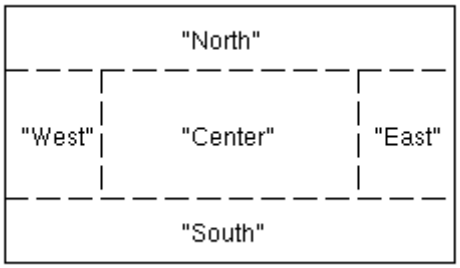
El panel de widgets:

```
Panel panel= new Panel();
```

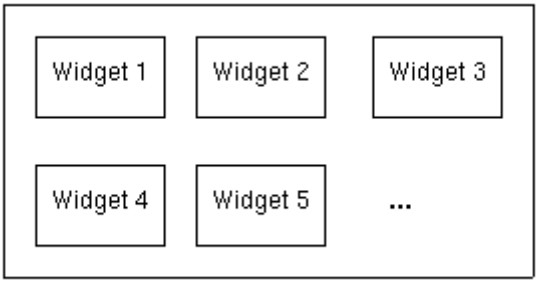
Organizador de contenedores: layout managers

- *No es conveniente asignar a los widget posiciones absolutas dentro de la ventana.*
- Para organizar el contenedor se le asigna un *layout manager* usando setLayout:
  - frame.setLayout(new BorderLayout());
  - panel.setLayout(new FlowLayout());
- El layout manager elige la mejor posición y tamaño de cada widget de acuerdo al espacio disponible.

BorderLayout organiza un contenedor en 5 zonas:



FlowLayout coloca los widgets de izquierda a derecha y de arriba hacia abajo:



Se colocan los widgets en los contenedores:

- Los botones y el campo de texto dentro del panel:
  - panel.add(texto);
  - panel.add(boton);
  - panel.add(check);
- La etiqueta, el área de texto y la lista en la ventana:
  - frame.add("North", etiq);
  - frame.add("West", area);
  - frame.add("East", lista);

El panel se trata como si fuese un solo widget:

frame.add("South", panel);

Finalmente se despliega la ventana con:
frame.pack();
frame.show();