élogos

APLICACIONES BASADAS EN ENTORNO GRÁFICO

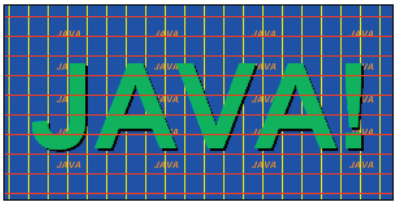
Applets

élogos

Índice

APLICACIONES BASADAS EN ENTORNO GRÁFICO

1. Características de un Applet. Ciclo de vida



Objetivos

1. Conocer qué es un Applet y cómo se crea.
2. Reconocer la inclusión de un Applet en una página web.
3. La clase Graphics: texto, fuente y color.

Presentación del Applet en Java

Lo primero que hay que hacer es diferenciar una aplicación de un Applet. Las aplicaciones Java funcionan por sí solas usando el interprete java.exe. Sin embargo, los Applets Java deben ejecutarse desde un browser (un navegador) del WWW. Los Applets son llamados desde documentos HTML en el navegador.

Debido a que los Applets viajan a todas partes, pudiendo portar virus que podrían dañar los sistemas de archivos de ordenadores clientes, la JVM define sobre los Applets algunas restricciones de seguridad (SANDBOX SECURITY MODEL).

Los Applets no pueden leer o escribir sobre el sistema de ficheros, excepto en directorios específicos.

No pueden comunicarse con otro servidor que no sea el que guarda el Applet. Esto a veces se puede configurar desde el browser.

Los Applets no pueden ejecutar programas del sistema de lectura.

Los Applets no pueden cargar programas nativos de la plataforma local.

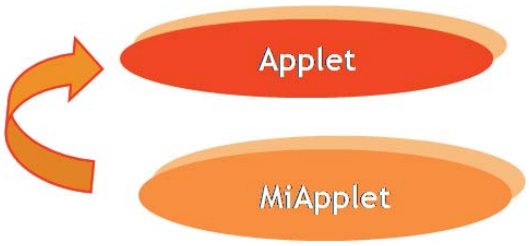
Creación de Applet

Para crear un Applet hemos de crear una subclase de la clase Applet que está dentro del paquete java.applet.

La sintaxis es:

```
import java.applet.Applet;
public class nombreClase extends  Applet {
...
}
```

La clase original debe ser pública.



Actividades del Applet

Para comenzar a trabajar con Applets, se necesita algo más que el main() usado en las aplicaciones. Un Applet necesita hacer algunas actividades asociadas a eventos importantes en el ciclo de vida del Applet y que están directamente relacionados con el funcionamiento del browser. Así, cada actividad tiene asociado su método.

Los principales métodos a tener en cuenta son:

1.- Inicialización:

Nada mas el Applet se carga. Entre lo que se puede hacer aquí hay que destacar:

a. Crear objetos que necesitemos.
b. Inicializar valores.
c. Cargar imágenes.
d. Cargar fuentes.

El método a sobreescribir (HOTPOT al pulsar aparece descripción inferior) es:

```
public void init() {
...
}
```

2.- Arranque:

Ocurre después de inicializar o después de una parada previa. Este proceso puede ocurrir muchas veces durante la vida de un Applet.

El método a sobreescribir (HOTPOT al pulsar aparece descripción inferior) es:

```
public void start() {
...
}
```

3.- Parada:

Ocurre cuando el usuario abandona la página que incluye este Applet o cuando llamamos nosotros a stop():

```
public void stop() {
...
}
```

4.- Destrucción:

Sirve para limpiar la memoria antes de abandónarla desde el navegador. Normalmente no sobreescribiremos este método a menos que queramos liberar recursos específicos.

La diferencia entre destroy() y finalize() está en que finalize() es mas genérico y sirve para un objeto cualquiera mientras que destroy() es propia del Applet.

El método a sobreescribir (HOTPOT al pulsar aparece descripción inferior) es:

```
public void destroy() {
...
}
```

Métodos del Applet

Otro de los principales métodos es el método paint().

Define cómo un Applet dibuja algo en la pantalla. Este proceso puede ocurrir cientos de veces en la vida del Applet.

El método a sobreescribir es:

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

Dentro del código de nuestra Applet hemos de incluir: import java.awt.Graphics;

El método paint (Gaphics g) recibe un objeto g de la clase Graphics. Este objeto se denomina contexto gráfico y se va a utilizar obligatoriamente siempre que queramos dibujar algo dentro del paint.

Ejemplo:

```
public void paint(Graphics g)
 {

 g.drawString ("Hola desde java", 10,10);
 }
```

En este ejemplo dibujamos un String (Hola desde java) en las coordenadas x=10 y=10 dentro del Applet. (Después se verán los métodos más importantes para dibujar en el Applet).

2. Inclusión de un Applet en una pagina web. Paso de parámetros

La etiqueta <APPLET>

Para incluir un Applet en una página web hay una etiqueta especifica; la etiqueta <APPLET>.

Dentro del cuerpo de nuestra pagina web y allí donde nos interese, pondríamos el Applet como:

```
<APPLET CODE="NombreApplet.class" WIDTH=200 HEIGHT=60>

Texto Alternativo

</APPLET>
```

- Detrás de la palabra CODE se pone entre comillas el nombre de la clase Applet que queremos. De este modo suponemos que el Applet (.class) se encuentra en el directorio donde está el documento HTML. En el nombre hay que incluir la extensión .class.

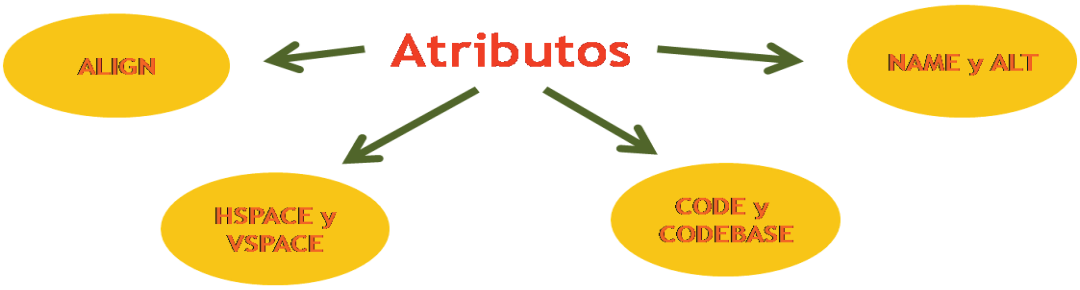
- El texto alternativo sirve para sustituir al Applet si el navegador no entiende la etiqueta <APPLET>. Es interesante hacer uso de esta opción para usuarios de nuestros Applets que no dispongan de un browser lo suficiente potente.

- WIDTH y HEIGHT serían respectivamente el ancho y el alto del applet en píxeles.

Atributos de la etiqueta <APPLET>

La etiqueta APPLET tiene varios atributos que se colocarían dentro de la misma etiqueta <APPLET ..... atributos> </APPLET>.

Los atributos son:



ALIGN

Define cómo debe alinearse el Applet. Puede tener 9 valores.
Para alinear en horizontal:

ALIGN=LEFT
Sitúa el Applet al margen izquierdo de la pagina web y todo el texto que sigue a la derecha del Applet.

ALIGN=RIGHT
Sitúa el Applet al margen derecho de la pagina web y todo el texto que sigue a la izquierda del Applet.

Si queremos que el texto siga debajo usaremos la etiqueta
 o su familia <BR CLEAR=opcion>, donde opción puede ser LEFT, RIGHT o ALL.

Para alinear en vertical usaremos:

ALIGN=TEXTTOP
Alinea la parte superior del Applet con la parte superior del texto en línea.

ALIGN=TOP
Alinea la parte superior del Applet con la parte superior del texto/imagen en línea.

ALIGN=ABSMIDDLE
Alinea el centro del Applet con el centro del texto/imagen en línea.

ALIGN=MIDDLE
Alinea el centro del Applet con la base del texto en línea.

ALIGN=BASELINE
Alinea la parte inferior del Applet con la base del texto en línea.

ALIGN=BOTTOM
Equivale a ALIGN=BASELINE.

ALIGN=ABSBOTTOM
Alinea la parte inferior del Applet con la parte más baja del texto/imagen/Applet en línea.

HSPACE y VSPACE
HSPACE Espacio horizontal en píxeles entre un Applet y el texto adjunto. Es el valor que deja tanto a izquierda como a derecha.

VSPACE Espacio vertical en píxeles entre un Applet y el texto adjunto. Es el valor que deja tanto arriba como abajo.

En definitiva, es como si la porción ocupada por el Applet fuera:

Anchura: HSPACE+WIDTH+HSPACE
Altura: VSPACE+HEIGHT+VSPACE
CODE y CODEBASE
CODE sirve para indicar la clase del Applet si ésta está en el mismo directorio que el archivo HTML en uso. CODEBASE es similar, pero dando la URL de donde se encuentra el Applet, bien absoluta o bien relativa, al archivo HTML actual. Es decir, haríamos:

<APPLET CODE="nombreClase.class" CODEBASE="clases"
WIDTH=100 HEIGHT=100> Java Applet Here </APPLET>

NAME y ALT

NAME sirve para nombrar el Applet en caso de tener varios y tener que trabajar con ellos.
ALT se utiliza opcionalmente para indicar un texto alternativo por si no se carga el Applet (es lo mismo que el texto sustitutivo antes comentado).

La etiqueta <EMBED>

Actualmente los únicos archivos ejecutables embebidos en documentos HTML son los Applets de Java y los controles Actives de Microsoft. Debido a que pronto surgirán nuevos ejecutables en nuevos lenguajes de programación, éstos para estar embebidos en los HTML tendrían una nueva etiqueta al igual que Java tiene <APPLET>.

Para unificar criterios se define una etiqueta genérica denominada <EMBED> para representar a cualquier recurso que pueda ser cargado dentro del documento HTML.

Ejemplo:

```
<APPLET CODE="nombreClase.class" CODEBASE="clases"
WIDTH=100 HEIGHT=100> Java Applet  </APPLET>
```

Utilizando la etiqueta <EMBED> escribiríamos:

```
<EMBED SCR="clases/nombreClase.class"
WIDTH=100 HEIGHT=100> Java Applet  </EMBED
```

La etiqueta <EMBED> hereda todas las propiedades que hasta ahora tenia la etiqueta <APPLET> y por ello tiene la misma sintaxis y los mismos atributos que ésta. Sólo cambia que los atributos CODE y CODEBASE se unen en un único atributo SCR.

Pasando parámetros a Applets

A los Applets también se les pueden pasar parámetros, para ello es necesario:

1. Una etiqueta especial en el archivo HTML.
2. Código en nuestro Applet para tratar estos parámetros.

Los parámetros en un Applet van en dos partes: un nombre y su valor. La etiqueta especial del lenguaje HTML es <PARAM>.

Su uso es:

```
<APPLET CODE="nombreClase" WIDTH=100 HEIGHT=100>
<PARAM NAME=”nombre” VALUE="Luis">
<PARAM NAME=”ciudad”   VALUE="Madrid">
...

</APPLET>
```

Pasando parámetros a Applets en su inicialización

Los parámetros se pasan al Applet en su inicialización en el método init() y accederemos al valor de los parámetros pasados al applet dentro del programa a través del método getParameter().

Este método requiere un argumento que es un String representando el nombre del parámetro y devuelve el valor del mismo que será otro String.

Si no nos interesara un String, haríamos una conversión de tipo. Así en nuestro caso, en init(), para coger los parámetros anteriores se haría:

```
public void init() {
   String nombre = getParameter("nombre");
   String  ciudad = getParameter("ciudad");
   ...
}
```

Si el parámetro no está especificado

Hemos de tener presente que el lenguaje es case-sensitive. Si el parámetro que esperamos no está especificado en el fichero HTML, el método getParameter() devuelve null, por ello es interesante y ante una respuesta null, que nuestro programa utilice un valor por defecto:

```
if (nombre == null) {
    nombre = "Luis";
}
```

Si por ejemplo queremos hacer una conversión de String a entero:

```
int tamaño;
String s = getParameter("tamaño");
if (s == null) {
    tamaño = 12; // valor por defecto
}
else { tamaño = Integer.parseInt(s); }
```

3. Ejecutar Applets. La clase Graphics

Desarrollando y ejecutando un Applet

Para desarrollar y ejecutar un Applet primeramente realizamos el fichero Java con el código del Applet.

Después de construir el fichero .java se compila como siempre con javac: C:> javac MiApplet.java.

Luego, construimos el archivo MiApplet.htm e insertamos la clase (fichero.class) en el atributo code.

Ejemplo

```
import  java.awt.*;
public class MiApplet extends Applet
{
public void init( )
{
setBackground (Color.red);//pone el fondo del applet en rojo
}
public void paint(Graphics g)
{
g.drawString("Hola desde java...",20,20); // Escribimos en el applet
}
}
```

MiApplet.htm:

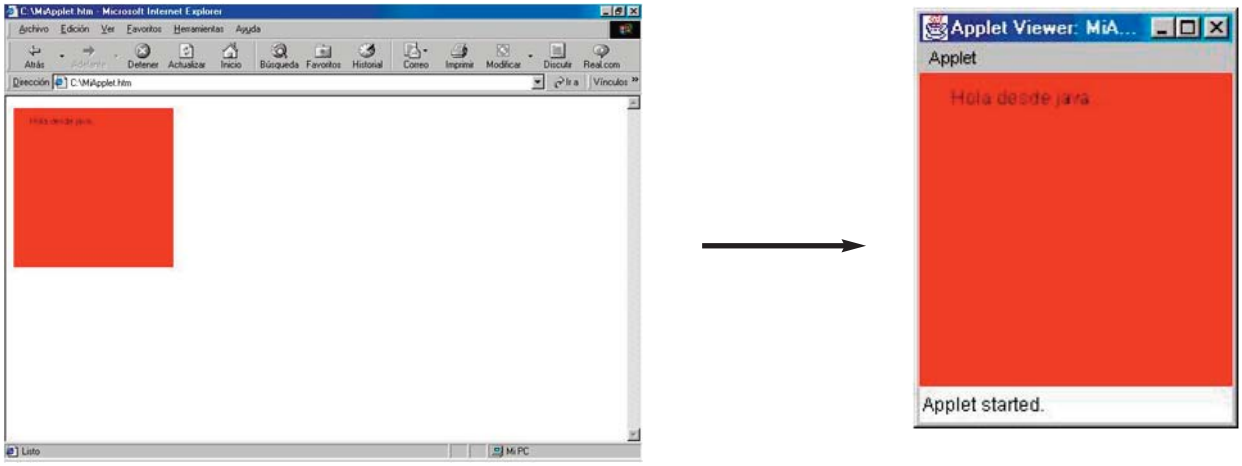
```
<html>
<body>
<applet code="MiApplet.class" width="200" height="200">
</applet>
</body>
</html>
```

Por último abrimos el Internet Explorer (browser) y abrimos la página MiApplet.htm con el resultado final.

Al cerrar este cuadro de texto, aparece con animación la flecha y la segunda imagen. Hot pot a esta segunda imagen, aparece lo siguiente:

También es posible poder visualizar el Applet sin un navegador, con la ayuda del appletviewer la herramienta de visualización de Applets incluida en el JDK.

Lo haríamos de la siguiente forma: C:>appletviewer MiApplet.htm



La clase Graphics

Para dibujar algo en nuestro Applet, no tenemos que crear siempre un objeto de esta clase, sino que vamos a hacer uso del objeto Graphics de la función paint(), y dibujando sobre este objeto, el resultado aparecerá en nuestro Applet.

Este objeto es pasado desde el navegador al propio Applet

Para hacer uso de la clase Graphics en nuestro Applet debemos incluir, al principio de nuestro Applet, como ya dijimos, el paquete que contiene a esta clase:

```
import java.awt.Graphics;
```

El origen de coordenadas del sistema de gráficos tiene su origen (0,0) en la esquina superior izquierda.

- Los valores positivos crecen hacia abajo y hacia la derecha.
- Los píxeles tienen asociados valores enteros, nunca fraccionarios.

Dibujar líneas

Para dibujar líneas basta usar el método drawLine(). Se le pasan 4 argumentos, las coordenadas del punto origen x1, y1 y las del punto destino x2 e y2.

Ejemplo

```
public void paint(Graphics g) {
   g.drawLine(x1,y1,x2,y2);
   ...
}
```

Existe la posibilidad de dibujar tres tipos de rectángulos. Para cada una de estas formas, disponemos de dos métodos, que lo dibuja y otro que lo dibuja y rellena. Los parámetros que se le pasan a estos métodos son las couno ordenadas (x,y) de la esquina superior izquierda, la anchura (w) y la altura (h).

- Rectángulos planos
- Rectángulos redondeados
- Rectángulos tridimensionales

Rectángulos planos:

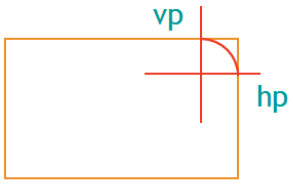
drawRect(x,y,w,h);	Lo dibuja
fillRect(x,y,w,h);	Lo dibuja y lo rellena

APLICACIONES BASADAS EN ENTORNO GRÁFICO

Rectángulos redondeados:

```
drawRoundRect(x,y,w,h,hp,vp);
fillRoundRect(x,y,w,h,hp,vp);
```

hp y **vp** marcan el tamaño del recorte en horizontal y vertical respectivamente, es decir la curvatura de las esquinas redondeadas

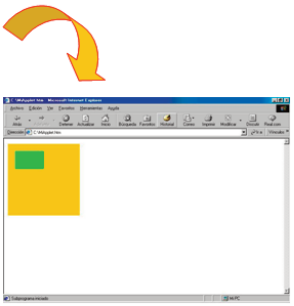


Rectángulos tridimensionales:

```
draw3DRect(x,y,w,h,eff);
fill3DRect(x,y,w,h,eff);
```

El argumento eff es un boolean que indica si el efecto del rectángulo es hacia afuera (true) o hacia dentro (false). Ejemplo:

```
import  java.applet.Applet;
import  java.awt.*;
public class MiApplet extends Applet
{
public void init( )
{
setBackground (Color.orange);//pone el fondo del applet en naranja
}
public void paint(Graphics g)
{
g.setColor(Color.green);//rellenamos d ecolr verde el rectángulo
g.fill3DRect(20,20,80,50,true);  // Escribimos en el applet
}
}
```



Dibujar un polígono

Para dibujar un polígono necesitamos de un set de puntos (x,y). El método dibuja líneas de punto a punto siguiendo el orden dado.

Como ocurría en los rectángulos, podemos dibujar y también rellenar:

```
drawPolygon();
fillPolygon();
```

El método drawPolygon() no cierra el polígono, así si queremos que se cierre hemos de incluir el primer punto como último punto también. El método fillPolygon() sí une el primer punto con el ultimo.

Los argumentos se le pueden pasar de dos formas dado que tenemos el método sobrecargado:

a. Tres argumentos: array con las coordenadas x, array con las coordenadas y, valor entero con el numero de puntos totales.
b. Un argumento: un objeto polígono, perteneciente a la clase Polygon.

Un objeto polígono se puede crear con la directiva new:

```
Polygon poli = new Polygon();
Polygon poli = new Polygon(vectorx,vectory,numPuntos);
```

y podemos añadirle puntos al polígono:

```
poli.addPoint(xNueva,yNueva);
```

APLICACIONES BASADAS EN ENTORNO GRÁFICO

Dibujar elipses y círculos

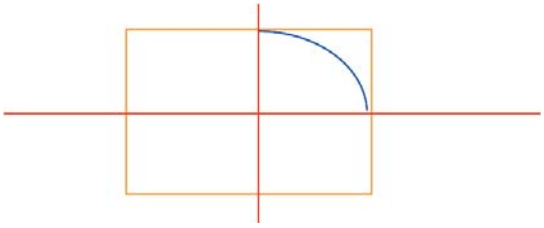
Para dibujar elipses y círculos, los parámetros que se le pasan vienen a ser los mismos que para dibujar rectángulos planos:

```
drawOval(x,y,ejehoriz,ejevert);
fillOval(x,y,ejehoriz,ejevert);
```

Se dibuja el óvalo que está circunscrito en el rectángulo que hemos definido.

Dibujar Arcos

Para dibujar arcos utilizaremos la siguiente función:

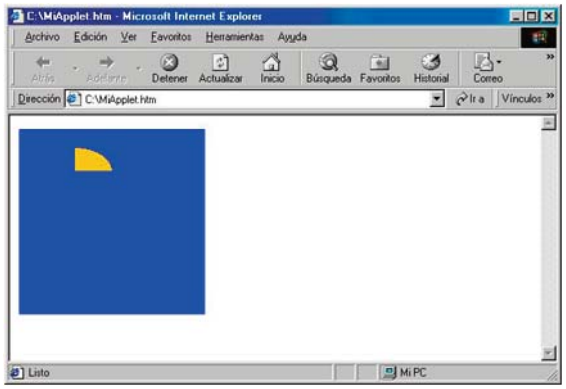


```
drawArc(x,y,w,h,ang,grad);
fillArc(x,y,w,h,ang,grad);
```

El método fillArc() rellena como si el arco se tratara de una porción de una tarta. Para contar los grados, debemos tomar una circunferencia de referencia y situar el valor 0 donde un reloj marca las 3. Los grados aumentan en sentido contrario a las agujas del reloj.

Ejemplo

```
import  java.applet.Applet;
import  java.awt.*;
public class MiApplet extends Applet
{
public void init( )
{
setBackground (Color.blue);//pone el fondo del applet en naranja
}
public void paint(Graphics g)
{
g.setColor(Color.orange);//rellenamos d ecolr verde el rectángulo
g.fillArc(20,20,80,50,0,90); // Escribimos en el applet
}
}
```



Gráficos

Finalmente y en relación con los gráficos, es posible copiar una porción de un gráfico o borrarlo.
Los métodos son:

```
copyArea(x,y,w,h,distx,disty);
clearRect(x,y,w,h);
```

Donde x,y,w,h son los parámetros del área rectangular y donde distx y disty son los movimientos en x e y hacia donde vamos a copiar el área dada. Es decir, las nuevas coordenadas de la esquina superior izquierda del rectángulo son:

```
xnueva = x + distx
ynueva = y + disty
```

El método clearRect() dibuja un rectángulo con el color de fondo. Si queremos borrar el Applet completo podemos hacer uso de la función size() de la siguiente forma:

```
g.clearRect(0,0,size().width,size().height);
size().width:
size().height:
```

Devuelve el ancho del Applet
Devuelve el alto del Applet

Texto y fuentes

Un objeto fuente se caracteriza por su nombre, su estilo y su número de puntos.

- Fuentes: "TimesRoman", "Courier", "Helvetica", ...
- Estilos: Font.PLAIN, Font.BOLD, Font.ITALIC, ...
- Tamaño: 12, 24, 36, ...

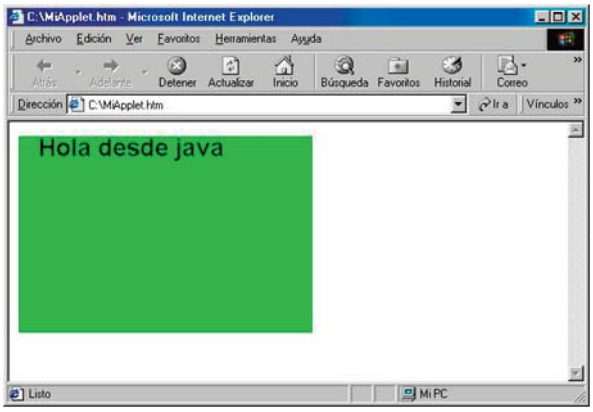
Los estilos se pueden combinar y crear un nuevo estilo:

Font.BOLD + Font.ITALIC	Texto en negrita y cursiva

Para usar una fuente debemos cargar el paquete: import java.awt.Font;

Para asignar la fuente al Applet usamos el método de la clase Graphics setFont() que recibe un objeto de tipo Font.

```
import java.awt.Font;
...
...
public void paint(Graphics g)
{
Font f=new Font("Arial",Font.BOLD,25);
g.setFont(f);    // Para asignar la fuente al Applet
g.drawString("Hola desde java",20,20);
}
}
```



Caracteres y Cadenas

Para dibujar caracteres y cadenas usaremos los métodos:

drawChars();
drawString();

Los dos valores numéricos “x” e “y” indican la posición donde comienza a dibujarse la cadena de caracteres.

El método drawChars funciona análogamente, pero se la pasa un array de caracteres. Tiene 5 argumentos:

g.drawChars(array,intFirst,intLast,x,y);

Los nuevos valores intFirst e intLast indican el primer y el último elemento del array que se va a dibujar.

- Existen una serie de métodos que nos proporcionan información sobre las fuentes

- Existe otra clase denominada FontMetrics que da información más especifica sobre una fuente. Para crear un objeto de este tipo haremos:

```
Font f = new Font("TimesRoman", Font.BOLD, 24);
FontMetrics fmetrics = getFontMetrics(f);
g.setFont(f);
```

Método	Objeto	Acción
getFont()	Graphics	devuelve el objeto fuente actual
getName()	Font	devuelve el nombre de la fuente: String
getSize()	Font	devuelve el tamaño de la fuente: int
getStyle()	Font	devuelve el estilo: 0 si es PLAIN 1 si es BOLD 2 si es ITALIC 3 si es BOLD ITALIC
isPlain()	Font	devuelve true si la fuente es plain
isBold()	Font	devuelve true si la fuente es bold
isItalic()	Font	devuelve true si la fuente es italic

Algunas de las funciones de esta nueva clase son:

Método	Significado
stringWidth(string)	devuelve la anchura total del string en píxeles
charWidth(char)	devuelve la anchura total del carácter en píxeles
getAscent()	devuelve la altura desde la base en píxeles
getDescent()	devuelve la altura desde la base a bajo en píxeles
getLeading()	devuelve la diferencia mínima entre dos líneas
getHeight()	altura total: descendiente mas ascendiente.

Ejemplo

```
public void paint(Graphics g) {
Font f = new Font("TimesRoman", Font.BOLD, 24);
g.setFont(f);
g.drawString("Esta es una fuente grande. ",x,y);
...
}
```

Color

Usando la clase Color podemos escribir y dibujar en varios colores. Para eso crearemos una instancia de la clase Color.

Esta clase define ciertos colores almacenados en variables pero si queremos otros colores de la paleta menos populares podemos hacerlo igualmente. Los colores más populares son:

Nombre	Valor RGB
Color.white	255,255,255
Color.black	0,0,0
Color.lightGray	192,192,192
Color.gray	128,128,128
Color.darkGray	64,64,64
Color.red	255,0,0
Color.green	0,255,0
Color.blue	0,0,255
Color.yellow	255,255,0
Color.magenta	255,0,255
Color.cyan	0,255,255
Color.pink	255,175,175
Color.orange	255,200,0

Si queremos cualquier otro color o de forma mas general haremos:

Color c = new Color(R,G,B); Los valores de R, G y B varían entre 0 y 255.

Método de la clase color

Para dibujar en un cierto color, en el método paint(), por ejemplo, haremos referencia a éste como sigue:

```
g.setColot(Color.green);
g.setColor(c);
```

Además podemos fijar los colores del fondo y del plano principal (texto e imágenes) utilizando los métodos:

```
setBackground(Color.white);
setForeground(Color.black);
```

Además de configurar los colores, podemos leer sus valores:

`getColor();`	Devuelve el color
`getBackground();`	Devuelve el color de fondo
`getForeground();`	Devuelve el color del primer plano

4. Resumen

Has llegado al final de este recurso formativo que denominamos “Applets”
En esta lección hemos estudiado los siguientes contenidos: