

Las clases *Color*, *Font* y *FontMetrics*

La clase *Color*

Los colores primarios son el rojo, el verde y el azul. Java utiliza un modelo de color denominado RGB, que significa que cualquier color se puede describir dando las cantidades de rojo (Red), verde (Green), y azul (Blue). Estas cantidades son números enteros comprendidos entre 0 y 255, o bien, números reales comprendidos entre 0.0 y 1.0. La siguiente tabla nos proporciona los colores más comunes y sus valores RGB.

Nombre	Red (rojo)	Green (verde)	Blue (azul)
white	255	255	255
lightGray	192	192	192
gray	128	128	128
drakGray	64	64	64
black	0	0	0
red	255	0	0
pink	255	175	175
orange	255	200	0
yellow	255	255	0
green	0	255	0
magenta	255	0	255
cyan	0	255	255
blue	0	0	255

Para crear un objeto de la clase *Color*, se pasan tres números a su constructor que indican la cantidad de rojo, verde y azul.

```
Color colorRosa=new Color(255, 175, 175);
```

Mediante la función *setColor*, cambiamos color con el que dibujamos una línea, un texto o rellenamos una figura cerrada en el contexto gráfico *g*.

```
g.setColor(colorRosa);
```

No es necesario tener a mano la tabla de las componentes RGB de cada color. La clase *Color* nos proporciona un conjunto de colores predefinidos en forma de miembros estáticos de dicha clase. Podemos escribir alternativamente

```
g.setColor(Color.pink);
```

Los colores predefinidos son los siguientes

Color.white Color.black Color.yellow
Color.lightGray Color.red Color.green
Color.gray Color.pink Color.magenta
Color.darkGray Color.orange Color.cyan
Color.blue

El color de fondo del componente se establece con *setBackground* y se obtiene con *getBackground*. En el siguiente applet observamos cómo se utiliza esta segunda función para crear una diana. En la función *init* establecemos el color de fondo en blanco mediante *setBackground*. En la función miembro *paint* obtenemos el color de fondo mediante *getBackground*. Los círculos se pintan de mayor a menor radio. Se pinta un círculo de color rojo y se borra parte de su interior con el color de fondo, de este modo se crea un anillo, luego otro y así sucesivamente, hasta completar cuatro anillos de color rojo con la apariencia de una diana.

En el programa, también podemos apreciar que la función *paint* suministra el contexto gráfico *g* del componente (applet) en el cual podemos dibujar. El objeto *g* llama a *setColor* para establecer el color, y a *fillOval* para dibujar un círculo pintado de dicho color.

Las función *getSize* nos devuelve el tamaño del componente (applet), de modo que los diámetros de la elipse mayor son respectivamente la anchura y altura del applet.

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

public class PaintApplet extends Applet {
    public void init() {
        setBackground(Color.white);
    }
    public void paint(Graphics g) {
        int x, y, ancho, alto;
        int appletAlto = getSize().height;
        int appletAncho = getSize().width;

        for (int i=8; i>=0; i--) {
            if ((i % 2)==0) g.setColor(Color.red);
            else g.setColor(getBackground());
            alto = appletAlto*i/8;
            ancho = appletAncho*i/8;
            x=appletAncho/2-i*appletAncho/16;
            y=appletAlto/2-i*appletAlto/16;
            g.fillOval(x, y, ancho, alto);
        }
    }
}
```

La clase *Font*

Para crear una fuente de texto u objeto de la clase *Font* llamamos a su constructor, y le pasamos el nombre de la fuente de texto, el estilo y el tamaño. Por ejemplo,

```
Font fuente=new Font("TimesRoman", Font.BOLD, 12);
```

Esta sentencia, crea una fuente de texto Times Roman, en letra negrita, de 12 puntos.

Los estilos vienen dados por constantes (miembros estáticos de la clase *Font*), *Font.BOLD* establece el estilo negrita, *Font.ITALIC*, el estilo cursiva, y *Font.PLAIN*, el estilo normal. Se pueden combinar las constantes *Font.BOLD+Font.ITALIC* para establecer el estilo negrita y cursiva a la vez.

La función *setFont* de la clase *Graphics* establece la fuente de texto en el contexto gráfico *g*.

```
g.setFont(fuente);
```

La función *getFont* obtiene la fuente de texto actual de dicho contexto gráfico. La función *drawString* dibuja el string guardado en el objeto *texto* de la clase *String*, y lo sitúa en la posición cuyas coordenadas vienen dadas por los dos números enteros que le siguen.

En la siguiente porción de código, establecemos una fuente de texto, dibujamos el texto, y reestablecemos la fuente de texto por defecto, una operación habitual que se realiza al programar un applet.

```
Font oldFont=getFont();
Font fuente=new Font("Monospaced", Font.BOLD, 36);
g.setFont(fuente);
g.drawString(texto, 100, 50);
g.setFont(oldFont);
g.drawString(otroTexto, 100, 70);
```

La clase *FontMetrics*

La clase *FontMetrics* nos permite conocer las características de una fuente de texto.

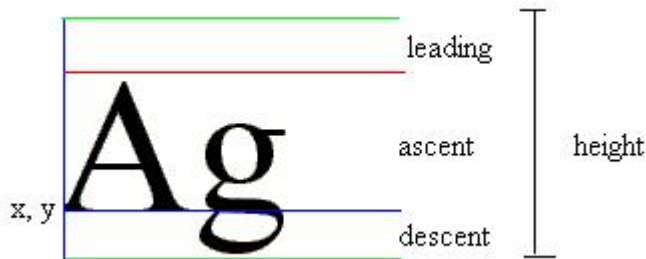
Desde el contexto gráfico *g*, llamamos a la función *getFontMetrics* para obtener un objeto *fm* de la clase *FontMetrics* que nos describe las características de una fuente determinada o de la fuente actualmente seleccionada. En el primer caso escribimos

```
Font fuente=new Font("Dialog", Font.BOLD, 36);
FontMetrics fm=g.getFontMetrics(fuente);
```

En el segundo caso, escribimos

```
Font fuente=new Font("Courier", Font.BOLD, 36);
g.setFont(fuente);
FontMetrics fm=g.getFontMetrics();
```

En el applet mostramos las características de una fuente de texto: *Ascent* es la distancia entre línea horizontal de color azul (baseline) y la línea horizontal de color rojo. *Descent* es la distancia entre la línea horizontal de color azul (baseline) y la línea horizontal de color verde. *Leading* sería la distancia entre las línea de color verde (descent) y la línea roja (ascent) de la siguiente línea de texto..



Para obtener la altura de una fuente de texto, llamamos a la función *getHeight* miembro de *FontMetrics*. La altura de una fuente de texto, es la distancia entre dos líneas base (baseline) consecutivas, y es la suma de el *ascent*, *descent* y *leading*.

Tres funciones que comienzan por *get* devuelven los valores de estos tres atributos de una fuente de texto: *getAscent*, *getDescent*, y *getLeading*.

Para escribir dos líneas de texto, una debajo de otra escribimos

```
FontMetrics fm=g.getFontMetrics();
String texto="La cigüeña vendrá";
g.drawString(texto, 10, 50);
int hFont=fm.getHeight();
texto=new String("serÁ en primavera");
g.drawString(texto, 10, 50+hFont);
```

La primera línea de texto, se sitúa en el punto (10, 50), la ordenada 50, señala la posición vertical de la línea base de la fuente de texto, véase la figura. La segunda línea de texto tiene una línea base cuya posición vertical se obtiene sumando a 50 la altura *hFont* de la fuente de texto.

Otro valor interesante, es la anchura de un texto, que se obtiene mediante la función miembro *stringWidth*, y se le pasa el texto. Por ejemplo, para centrar horizontalmente un texto en el applet escribimos.

```
String texto="La cigüeña vendrá";
int ancho=fm.stringWidth(texto);
g.drawString(texto, (anchoApplet-ancho)/2, 50);
```

La función *getSize().width* obtiene la anchura del componente (applet), y la variable *ancho*, guarda la anchura del string *texto*.

El código completo de este ejemplo es, el siguiente

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

```
public class FontApplet2 extends Applet {
    public void init() {
        setBackground(Color.white);
    }
    public void paint(Graphics g){
        int anchoApplet=getSize().width;
        Font oldFont=getFont();
        Font fuente=new Font("Monospaced", Font.BOLD, 36);
        g.setFont(fuente);
        FontMetrics fm=g.getFontMetrics();
        String texto="La cigüeña vendrá";
        int ancho=fm.stringWidth(texto);
        int y=50;
        //texto centrado
        g.drawString(texto, (anchoApplet-ancho)/2, y);
        texto=new String("serÁ en primavera");
        ancho=fm.stringWidth(texto);
        int hFont=fm.getHeight();
        g.drawString(texto, (anchoApplet-ancho)/2, y+hFont);
    }
}
```