# 1) Color

- **Librería:** *Java 2D*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.awt*

La clase Color representa un color en formato RGB o RGBA, siendo R la cantidad de rojo, G la cantidad de verde, B la cantidad de azul y A el valor de transparencia (o canal alfa). Estos valores deben estar entre 0 y 255.

| Color |
|---|
| |
| + Color (int r, int g, int b)<br>+ Color (int r, int g, int b, int a)<br>+ int getRed()<br>+ int getGreen()<br>+ int getBlue()<br>+ int getAlpha()<br>+ Color brighter()<br>+ Color darker() |

- El primer constructor crea un color en formato RGB
- El segundo constructor crea un color en formato RGBA
- Los métodos get devuelven los valores R,G,B y A del color
- brighter: Devuelve un nuevo color más claro que el actual
- darker: Devuelve un nuevo color más oscuro que el actual

Esta clase define constantes con los colores habituales predefinidos:

| Constante | Tipo de dato | Valor |
|---|---|---|
| Color.RED | Color | El color rojo |
| Color.BLACK | Color | El color negro |
| Color.WHITE | Color | El color blanco |
| Color.GREEN | Color | El color verde |
| Color.BLUE | Color | El color azul |
| Color.YELLOW | Color | El color amarillo |
| Color.ORANGE | Color | El color naranja |
| Color.GRAY | Color | El color gris |
| Color.MAGENTA | Color | El color magenta |

# 2) Image

- **Librería:** *Java 2D*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.awt*

La clase Image representa una imagen cualquiera, que puede proceder de muy diversas fuentes: fotos, escáner, gráficos, etc.

| Image |
| --- |
| |
| + Graphics getGraphics()<br>+ int getWidth(ImageObserver o)<br>+ int getHeight(ImageObserver o)<br>+ Image getScaledInstance(int w, int h, int m) |

- getGraphics: Obtiene un objeto con el que podemos dibujar sobre la imagen.
- getWidth: Devuelve el ancho de la imagen. Debe pasarse null como parámetro.
- getHeight: Devuelve la altura de la imagen. Debe pasarse null como parámetro.
- getScaledInstance: Devuelve un objeto imagen con una copia escalada de la imagen original. Los parámetros de este método son:
    - w: La anchura de la imagen escalada.
    - h: La altura de la imagen escalada.
    - m: Una constante que indica el tipo de calidad del proceso de escalado:

| Constante | Tipo de dato | Valor |
| --- | --- | --- |
| Image.FAST | int | Realiza un escalado rápido, aunque de mala calidad |
| Image.SMOOTH | int | Realiza un escalado lento, de buena calidad |

Es importante destacar que cuando se llama al método getScaledInstance la imagen se escala en segundo plano y no está disponible hasta que pasan unos segundos. Para que el programa "espere" a que la imagen escalada esté disponible puede hacerse una pausa con el método Thread.sleep (aunque se corre el riesgo de no esperar lo suficiente o colarnos), o usar la clase **MediaTracker**, que permite parar el programa hasta que la imagen esté lista.

# 3) Font

- **Librería:** *Java 2D*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.awt*

La clase Font representa un tipo de letra. Lo más corriente es usar tipos de letra en formato True Type Font (TTF) o Postscript Type-1 de Adobe (PFB). Los tipos de letra pueden estar instalados en el sistema operativo, o guardados como archivos.

| Font |
|---|
| |
| + static Font createFont(int tipo, File archivo) throws Exception<br>+ static Font decode (String n) throws Exception<br>+ String getName()<br>+ String getFamily()<br>+ Font deriveFont(float tam)<br>+ Font deriveFont(int estilo) |

- createFont: Nos permite obtener un tipo de letra a partir de un objeto File con la ruta de un archivo. El primer parámetro indica por medio de una constante el formato del tipo de letra incluido en el archivo:

| Constante | Tipo de dato | Valor |
|---|---|---|
| Font.TRUETYPE_FONT | Int | Formato de tipo de letra True Type |
| Font.TYPE1_FONT | int | Formato de letra Postscript Type 1 |

- decode: Obtiene un tipo de letra instalado en el sistema operativo. Debe usarse un string con el nombre deseado, o una de las siguientes constantes para obtener un tipo de letra por defecto dentro de una familia predefinida:

| Constante | Tipo de dato | Valor |
|---|---|---|
| Font.MONOSPACED | String | Familia de letras de igual anchura (monospace) |
| Font.SANS_SERIF | int | Familia de letra Sans Serif |
| Font.SERIF | Int | Familia de letra Serif |

- getName: Devuelve el nombre del tipo de letra.
- getFamily: Devuelve el nombre de la familia del tipo de letra.
- primer deriveFont: Convierte el tipo de letra al tamaño indicado.
- segundo deriveFont: Convierte el tipo de letra al estilo indicado, según estas constantes:

| Constante | Tipo de dato | Valor |
|---|---|---|
| Font.PLAIN | int | Tipo de letra "plano", es decir, normal |
| Font.BOLD | int | Tipo de letra en negrita |
| Font.ITALIC | Int | Tipo de letra cursiva |

# 4) Graphics

- **Librería:** *Java 2D*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.awt*

La clase Graphics es muy importante en Java, ya que permite dibujar sobre el objeto que nos lo proporciona. Con Graphics, se puede dibujar de la misma forma en la pantalla, en una imagen, en una página de un archivo PDF, la impresora, etc.

| Graphics |
| --- |
| |
| + void setColor (Color c)<br>+ void setFont (Font f)<br>+ void drawLine(int x0, int y0, int x1, int y1)<br>+ void drawRect(int x, int y, int w, int h)<br>+ void fillRect(int x, int y, int w, int h)<br>+ void draw3DRect(int x, int y, int w, int h, boolean elevado)<br>+ void drawRoundRect (int x, int y, int w, int h, int aw, int ah)<br>+ void drawOval (int x, int y, int w, int h)<br>+ void fillOval (int x, int y, int w, int h)<br>+ void drawString (String s, int x, int y)<br>+ void drawImage (Image i, int x, int y, ImageObserver o)<br>+ void drawImage (Image i, int x, int y, int w, int h, ImageObserver o)<br>+ void drawPolygon (int[] x, int[] y, int n)<br>+ void dispose() |

- setColor: Establece el color que se usará para todas las operaciones de dibujado.
- setFont: Establece el tipo de letra que se usará para dibujar texto.
- drawLine: Dibuja la línea que une el punto de coordenadas (x0,y0) con (x1,y1)
- drawRect: Dibuja un rectángulo con vértice superior izquierdo (x,y), ancho w y alto h
- fillRect: Es similar a drawRect, pero rellena de color el rectángulo.
- draw3DRect: Dibuja un rectángulo con efecto tridimensional. El parámetro de elevación indica si deberá dibujarse elevado o hundido.
- drawRoundRect: Dibuja un rectángulo de esquinas redondeadas. Los parámetros aw y ah definen la anchura y altura del arco de redondeo.
- drawOval: Dibuja una elipse con esquina superior izquierda (x,y). Los parámetros w y h indican la anchura y altura de la elipse. Si w=h, se obtiene un círculo.
- fillOval: Es como el anterior, pero rellenando de color la elipse/círculo.
- drawString: Dibuja una cadena de texto cuya línea base está en las coordenadas (x,y)
- primer drawImage: Dibuja una imagen rectangular poniendo su vértice superior izquierda en las coordenadas (x,y).
- segundo drawImage: Es igual que el primer drawImage, pero escala la imagen de forma que la anchura sea w y la altura h.
- drawPolygon: Dibuja un polígono de n lados en la pantalla cuyas coordenadas $(x_1,y_1),...(x_n,y_n)$ se encuentran definidas en los arrays indicados.
- dispose: Libera los recursos del objeto Graphics. Debe usarse cuando ya no se vaya a utilizar más dicho objeto.

# 5) Point y Dimension

- **Librería:** *Java 2D*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.awt*

Las clases Point y Dimension son muy parecidas y se utilizan para indicar un punto de la pantalla expresado en coordenadas (x,y) y unas dimensiones expresadas en formato (w,h) siendo w anchura y h altura.

| Point |
| --- |
| |
| + Point (int x, int y)<br>+ double getX()<br>+ double getY() |

- El constructor crea un objeto Point para representar el punto de coordenadas (x,y)
- getX: Devuelve la coordenada X del punto
- getY: Devuelve la coordenada Y del punto

| Dimension |
| --- |
| |
| + Dimension (int w, int h)<br>+ double getWidth()<br>+ double getHeight() |

- El constructor crea un objeto Dimension para representar una medida de anchura w y altura h
- getWidth: Devuelve la anchura de la medida expresada en el objeto.
- getHeight: Devuelve la altura de la medida que expresa el objeto.

# 6) Rectangle

- **Librería:** *Java 2D*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.awt*

La clase Rectangle representa un rectángulo que se define mediante las coordenadas de su esquina superior izquierda, su anchura y su altura.

| Rectangle |
| --- |
| |
| + Rectangle (int x, int y, int w, int h)<br>+ Rectangle (Point p, Dimension d)<br>+ double getX()<br>+ double getY()<br>+ double getWidth()<br>+ double getHeight()<br>+ void add (Point p)<br>+ void grow (int h, int v)<br>+ boolean contains (int x, int y)<br>+ boolean contains (Point p)<br>+ void setBounds (int x, int y, int w, int h)<br>+ void setLocation (Point p)<br>+ void setSize (Dimension d) |

- El primer constructor crea un rectángulo con vértice superior izquierdo en las coordenadas (x,y), anchura w y altura h
- El segundo constructor crea un rectángulo con vértice superior izquierdo especificado en el objeto p y su tamaño indicado en el objeto d.
- getX devuelve la coordenada X del vértice superior izquierdo del rectángulo
- getY devuelve la coordenada Y del vértice superior izquierdo del rectángulo
- getWidth devuelve la anchura del rectángulo
- getHeight devuelve la altura del rectángulo
- add: Añade un punto al rectángulo. Esto hace que el rectángulo crezca hasta contener dicho punto.
- grow: El rectángulo aumento su tamaño la cantidad horizontal h y la cantidad vertical v
- El primer método contains devuelve true si el punto de coordenadas (x,y) está en el rectángulo.
- El segundo método contains devuelve true si el punto indicado en el objeto p está en el rectángulo.
- setBounds: Cambia la posición y tamaño del rectángulo.
- setLocation: Cambia la posición del rectángulo.
- setSize: Cambia el tamaño del rectángulo.

# 7) Toolkit

- **Librería:** *Java 2D*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.awt*

Toolkit es una clase auxiliar que sirve para realizar algunas operaciones típicas con gráficos.

| Toolkit |
| --- |
| |
| + static Toolkit getDefaultToolkit()<br>+ Cursor createCustomCursor(Image i, Point p, String n)<br>+ void beep()<br>+ int getScreenResolution()<br>+ Dimension getScreenSize() |

- El método getDefaultToolkit devuelve un objeto de la clase Toolkit

- createCustomCursor nos permite crear un objeto que puede ser usado como cursor de ratón personalizado. Recibe la imagen del cursor, el punto de la imagen que se considera como punto de "clic" y el nombre que se le da al cursor.

- El método beep hace que se emita un sonido predeterminado por el sistema operativo.

- getScreenResolution devuelve la resolución de la pantalla, en bpp (bits por píxel)

- getScreenSize devuelve un objeto con el tamaño de la pantalla en formato (ancho, alto).

# 8) ImageIO

- **Librería:** *Java 2D*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *javax.imageio*

ImageIO es una clase cuyos métodos nos permiten cargar y guardar imágenes en una amplia variedad de formatos.

| ImageIO |
| --- |
| |
| + static BufferedImage read (File f)<br>+ static BufferedImage read (InputStream a)<br>+ static boolean write (BufferedImage img, String formato, File f)<br>+ static boolean write (BufferedImage img, String formato, OutputStream a)<br>+ static String[] getReaderFormatNames()<br>+ static String[] getWriterFormatNames() |

- El primer método read carga una imagen cuya ruta se pasa como parámetro y la devuelve en forma de BufferedImage.
- El segundo método read es igual que el anterior, pero lee la imagen desde un canal de entrada arbitrario, no necesariamente un archivo.
- El primer método write guarda una imagen al formato cuyo nombre se pasa como segundo parámetro. La lista de formatos admitidos depende del sistema operativo, pero se admiten los siguientes: **jpeg, png, bmp, gif** (algunas máquinas virtuales pueden admitir más). El tercer parámetro es el archivo donde se guardará la imagen.
- El segundo método write es como el anterior, pero escribe la imagen en un canal de salida arbitrario, no necesariamente un archivo.
- El método getReaderFormatNames devuelve la lista de formatos de lectura soportados por la máquina virtual.
- El método getWriterFormatNames devuelve la lista de formatos de escritura soportados por la máquina virtual.

# 9) BufferedImage

- **Librería:** *Java 2D*
- **Archivo:** *Incluido en el JDK*
- **Desarrollador:** *Sun Microsystems*
- **Paquete:** *java.awt*

Una BufferedImage es una imagen guardada en memoria (no se ve hasta que el programador la utilice en algún sitio) sobre la que el usuario puede dibujar libremente.

| BufferedImage extends Image |
|---|
| |
| + BufferedImage (int alto, int ancho, int tipoImagen)<br>+ Graphics getGraphics()<br>+ int getWidth()<br>+ int getHeight()<br>+ int getRGB (int x, int y)<br>+ void setRGB (int x, int y, int rgb)<br>+ BufferedImage getSubimage (int x, int y, int w, int h) |

- El constructor crea una imagen con la altura, anchura y tipo indicados en los parámetros. El tipo hace referencia a la forma en la que se representan los píxeles de la imagen, y puede ser una de estas constantes (hay muchas más):

| | |
|---|---|
| BufferedImage.TYPE_INT_ARGB | La imagen usa píxeles de 32 bits en formato alfa-red-green-blue |
| BufferedImage.TYPE_INT_RGB | La imagen usa píxeles de 24 bits en formato reg-green-blue |
| BufferedImage.TYPE_BYTE_GRAY | La imagen usa escala de grises |
| BufferedImage.TYPE_BYTE_INDEXED | La imagen usa una paleta de colores |

- El método getGraphics obtiene un objeto Graphics para dibujar sobre la imagen
- El método getWidth devuelve la anchura en píxeles de la imagen
- El método getHeight devuelve la altura en píxeles de la imagen
- El método getRGB devuelve el color que hay en el píxel cuyas coordenadas se indican. Dicho color se encuentra empaquetado en los bytes del entero que devuelve el método.
- El método setRGB cambia el color del píxel que se pasa como parámetro. El nuevo color es empaquetado en los bytes del entero que se pasa como tercer parámetro.
- El método getSubimage extrae la imagen cuyo rectángulo tiene el vértice superior izquierdo en las coordenadas (x,y), anchura w y altura h.