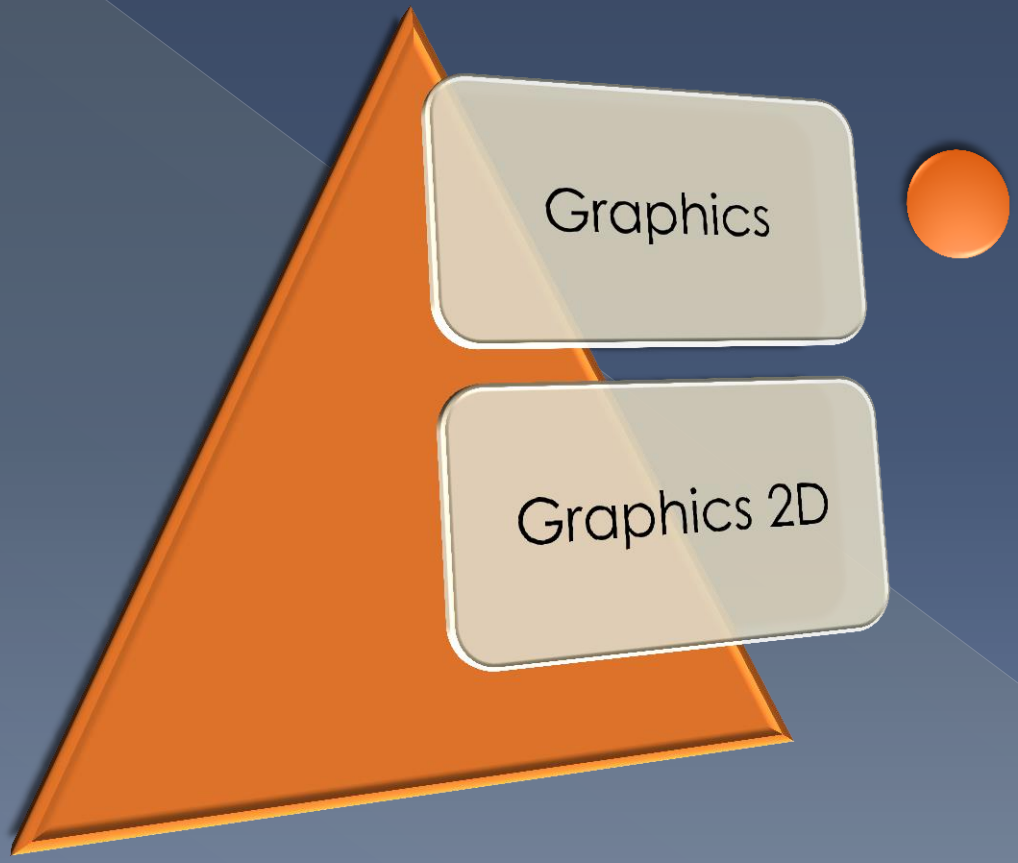
JAVA 2D Graphics

Índice



Clase "Graphics"

- *El primer enfoque (JDK 1.0) que abordaba la gestión de gráficos se basaba en el uso de la clase Graphics y sus métodos*

Pintar

```
drawLine(int x1,int y, int x2, int y2);
drawRect(int x1,int y, int w, int h);
drawOval(int x1,int y, int w, int h);
fillRect(int x1,int y, int w, int h);
fillOval(int x1,int y, int w, int h);
…
drawString(String s , int x, int y);
drawImage(Image  i , int x, int y,…);
```

Propiedades

```
setColor(Color c);
setFont(Font f);
```

Clase "Graphics"

No hay clases asociados a las formas geométricas

- *No se pueden editar las figuras ya pintadas*

Pocos atributos (color y fuente)

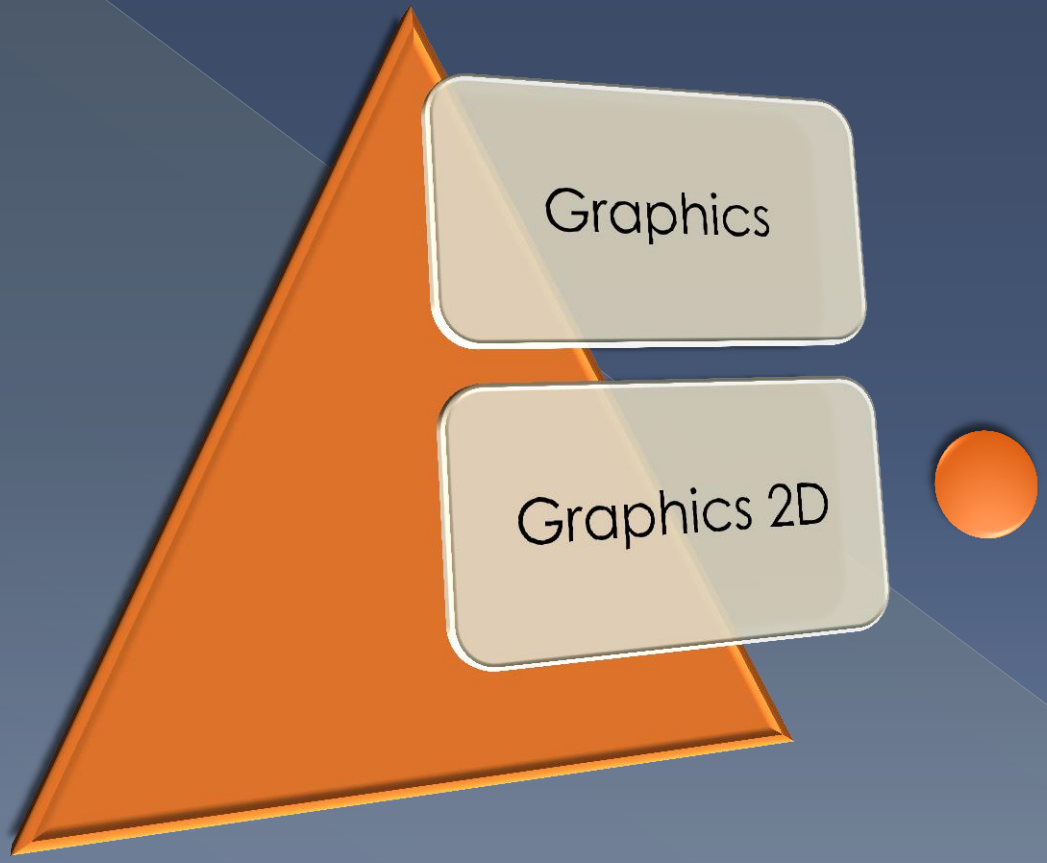
Pintar

```
drawLine(int x1,int y, int x2, int y2);
drawRect(int x1,int y, int w, int h);
drawOval(int x1,int y, int w, int h);
fillRect(int x1,int y, int w, int h);
fillOval(int x1,int y, int w, int h);
…
drawString(String s , int x, int y);
drawImage(Image  i , int x, int y,…);
```

Propiedades

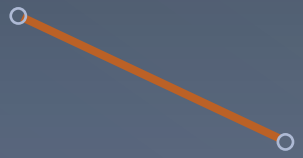
```
setColor(Color c);
setFont(Font f);
```

Índice



JAVA 2D Graphics

- *Mejora (sustancialmente) el enfoque basado en la clase Graphics*
  - *Introduce clases asociados a las formas geométricas*
  - *Incorpora nuevas formas y propiedades*

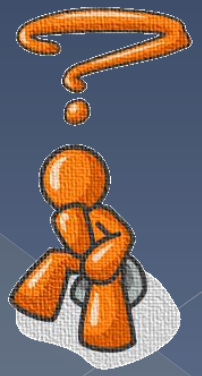


Punto de inicio
Punto de fin

Información geométrica

Color
Grosor
Discontinuidad
…

Atributos de la forma



JAVA 2D Graphics

- *Mejora (sustancialmente) el enfoque basado en la clase Graphics*
  - *Introduce clases asociados a las formas geométricas*
  - *Incorpora nuevas formas y propiedades*

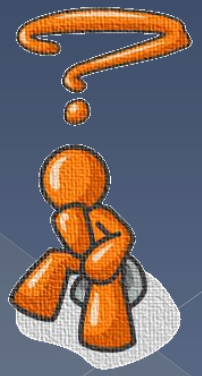
```
public class Linea ... {
  Point pIni,pFin;
  Color color;
  int grosor;
  .
  .
}
```

Punto de inicio
Punto de fin

Información geométrica

Color
Grosor
Discontinuidad
…

Atributos de la forma



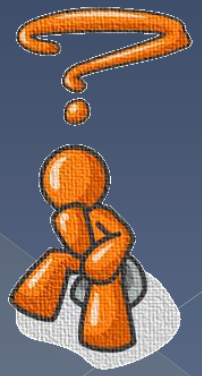
JAVA 2D Graphics

- *Mejora (sustancialmente) el enfoque basado en la clase Graphics*
  - *Introduce clases asociados a las formas geométricas*
  - *Incorpora nuevas formas y propiedades*
- *Distingue entre:*
  - *Forma: Representa la geometría de la figura* → Shape
  - *Contexto: Atributos con los que se pintará* → Graphics2D

```
public class Linea ... {
  Point pIni,pFin;
  Color color;
  int grosor;
  .
  .
}
```

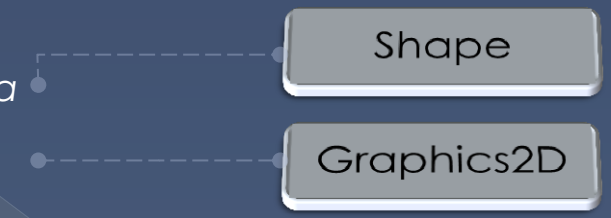
Punto de inicio
Punto de fin
Información geométrica

Color
Grosor
Discontinuidad
...
Atributos de la forma

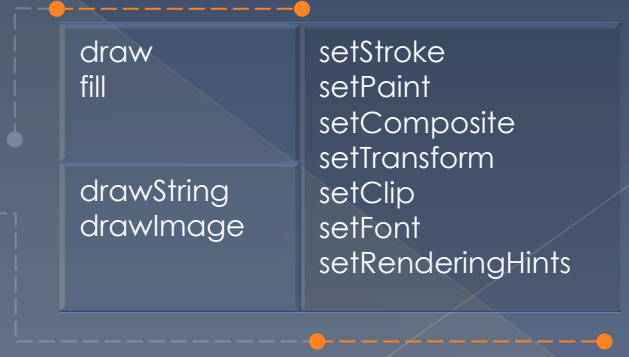


JAVA 2D Graphics

- *Mejora (sustancialmente) el enfoque basado en la clase Graphics*
  - *Introduce clases asociados a las formas geométricas*
  - *Incorpora nuevas formas y propiedades*
- *Distingue entre:*
  - *Forma: Representa la geometría de la figura*
  - *Contexto: Atributos con los que se pintará*



- *Incorpora una nueva clase Graphics2D*
  - *Hereda de Graphics*
  - *Métodos para dibujar formas, imágenes o texto*
  - *Métodos para cambiar atributos*
  - *Ambos aceptan objetos como parámetros*



JAVA 2D Graphics

Cuando usábamos Graphics…

```
public void  paint(Graphics g){
   super.paint(g);


   g.----------(--);

}
```

- *Incorpora una nueva clase Graphics2D*
  - *Hereda de Graphics*
  - *Métodos para dibujar formas, imágenes o texto*
  - *Métodos para cambiar atributos*
  - *Ambos aceptan objetos como parámetros*

draw fill	setStroke setPaint setComposite setTransform
drawString drawImage	setClip setFont setRenderingHints

JAVA 2D Graphics

Cuando usábamos Graphics…

```
public void  paint(Graphics g){
   super.paint(g);

   g.----------(--);

}
```

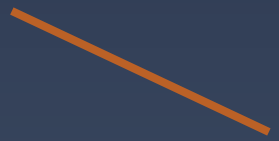
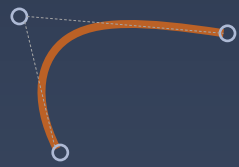
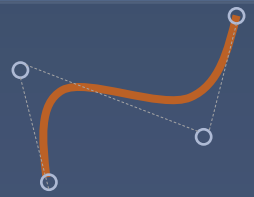
… ahora con Graphics2D

```
public void  paint(Graphics g){
   super.paint(g);
   Graphics2D g2d = (Graphics2D)g;

   g2d.----------(--);

}
```

- *Incorpora una nueva clase Graphics2D*
  - *Hereda de Graphics*
  - *Métodos para dibujar formas, imágenes o texto*
  - *Métodos para cambiar atributos*
  - *Ambos aceptan objetos como parámetros*

draw fill	drawString drawImage
setStroke setPaint setComposite setTransform	setClip setFont setRenderingHints

Formas: la clase Shape

Line2D		*QuadCurve2D*	
Rectangle2D		*CubicCurve2D*	
RoundRectangle2D		*GeneralPath*	
Ellipse2D		*Polygon*	
Arc2D		*Area*	Unión Intersección Diferencia XOR 

Formas: la clase Shape

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    Line2D linea = new Line2D.Float(new Point(20,20), new Point(70,70));
    g2d.draw(linea);

}
```

Formas: la clase Shape

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    g2d.draw(linea);
}

public void  mousePressed(MouseEvent evt){
    linea = new Line2D.Float(evt.getPoint(), evt.getPoint());
}

public void  mouseDraggued(MouseEvent evt){
    linea.setLine(line.getP1(), evt.getPoint());
    this.repaint();
}



Line2D linea;
```

3 Dibuja el objeto

1 Crea el objeto

2 Modifica el objeto

Formas: la clase Shape

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    g2d.setPaint(Color.red);
    for(Shape s:vShape){
       g2d.draw(s);
    }
}

public void  mousePressed(MouseEvent evt){
    linea = new Line2D.Float(evt.getPoint(), evt.getPoint());
    vShape.add(linea);
}

public void  mouseDraggued(MouseEvent evt){
    linea.setLine(line.getP1(), evt.getPoint());
    this.repaint();
}


Line2D linea;
List<Shape> vShape = new ArrayList();
```

Contexto

TRAZO

RELLENO

COMPOSICIÓN

TRANSFORMACIÓN

RECORTE

FUENTE

RENDERING

Contexto

Atributo que se aplica al contorno de una forma. Permite dibujar líneas con cualquier tamaño de punto y patrón de guiones, así como aplicar estilos de fin de línea y unión de línea.

setStroke(..)

El atributo de relleno se aplica al interior de una forma. Permite rellenar formas con colores sólidos, degradados y patrones.

setPaint(…)

El atributo de composición se utiliza cuando los objetos renderizados se superponen a los objetos existentes. Define las reglas de composición de unos y otros.

setComposite(…)

El atributo de transformación se aplica durante el renderizado para convertir las coordenadas de los objetos antes de ser mostrados. Incluye transformaciones de traslación, rotación y escala.

setTransorm(…)

El tipo de clip restringe el renderizado al área dentro del contorno del objeto Shape utilizado para definir el trazado de recorte.

setClip(…)

Fuente usada para mostrar texto

setFont(…)

Permite activar mejoras en el proceso de renderizado (trade-off entre velocidad y calidad).

setRenderingHints(…)

Contexto

TRAZO

RELLENO

COMPOSICIÓN

TRANSFORMACIÓN

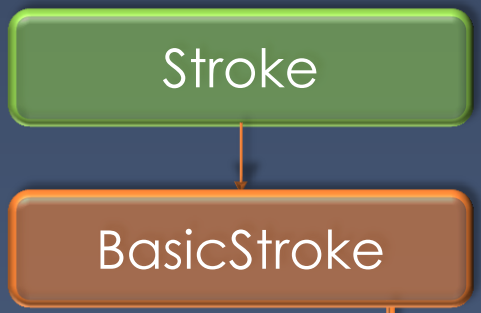
RECORTE

FUENTE

RENDERING

Trazo (Stroke)

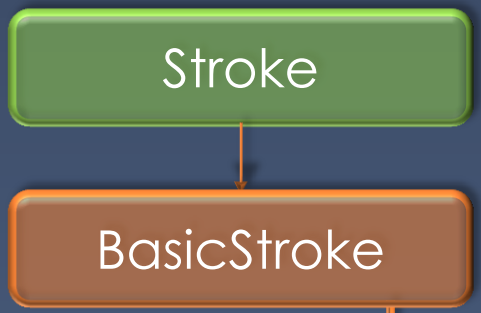
- *Establece el estilo del trazo y se aplica al contorno de una forma*
  - *Grosor*
  - *Estilo final de líneas*
  - *Estilo de unión de líneas*
  - *Discontinuidad*



BasicStroke()
BasicStroke(float width)
BasicStroke(float width, int cap, int join)
BasicStroke(float width, int cap, int join, float miterlimit)
BasicStroke(float width, int cap, int join, float miterlimit,
float[] dash, float dash_phase)

Trazo (Stroke)

- *Establece el estilo del trazo y se aplica al contorno de una forma*
  - *Grosor*
  - *Estilo final de líneas*
  - *Estilo de unión de líneas*
  - *Discontinuidad*



BasicStroke()
BasicStroke(float width)
BasicStroke(float width, int cap, int join)
BasicStroke(float width, int cap, int join, float miterlimit)
BasicStroke(float width, int cap, int join, float miterlimit,
float[] dash, float dash_phase)

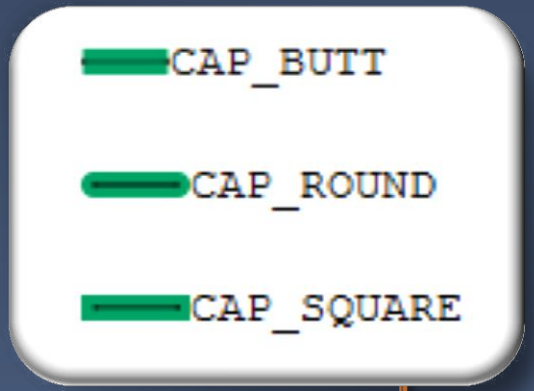
Trazo (Stroke)

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    Stroke trazo = new BasicStroke(10.0f);
    g2d.setStroke(trazo);
    g2d.draw(linea);
    g2d.draw(rectangulo);
    .
    .
}
```

```
Line2D linea;
Rectangle2D rectangulo;
```

Trazo (Stroke)

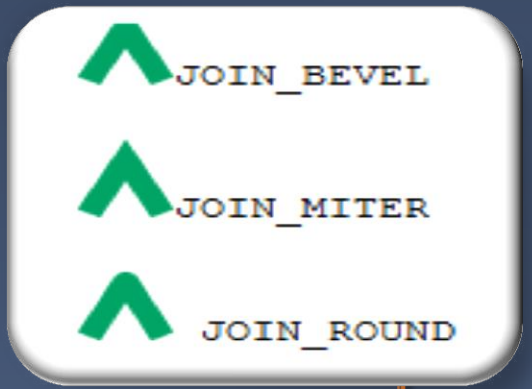
- *Establece el estilo del trazo y se aplica al contorno de una forma*
  - *Grosor*
  - *Estilo final de líneas*
  - *Estilo de unión de líneas*
  - *Discontinuidad*



BasicStroke()
BasicStroke(float width)
BasicStroke(float width, int cap, int join)
BasicStroke(float width, int cap, int join, float miterlimit)
BasicStroke(float width, int cap, int join, float miterlimit,
float[] dash, float dash_phase)

Trazo (Stroke)

- *Establece el estilo del trazo y se aplica al contorno de una forma*
  - *Grosor*
  - *Estilo final de líneas*
  - *Estilo de unión de líneas*
  - *Discontinuidad*



BasicStroke()
BasicStroke(float width)
BasicStroke(float width, int cap, int join)
BasicStroke(float width, int cap, int join, float miterlimit)
BasicStroke(float width, int cap, int join, float miterlimit,
float[] dash, float dash_phase)

Trazo (Stroke)

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    Stroke trazo = new BasicStroke(10.0f,BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND);
    g2d.setStroke(trazo);
    g2d.draw(linea);
    g2d.draw(rectangulo);
    .
    .
}
```

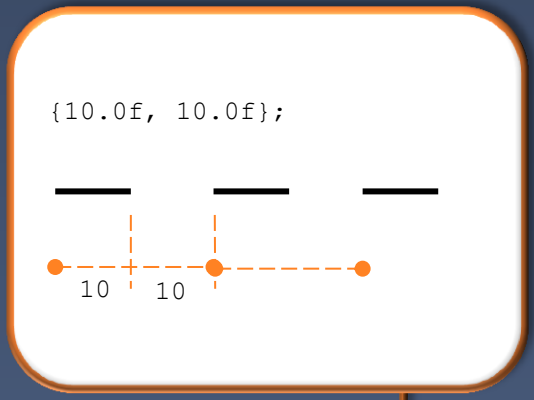
Estilo fin de línea

Estilo unión de línea

```
Line2D linea;
Rectangle2D rectangulo;
```

Trazo (Stroke)

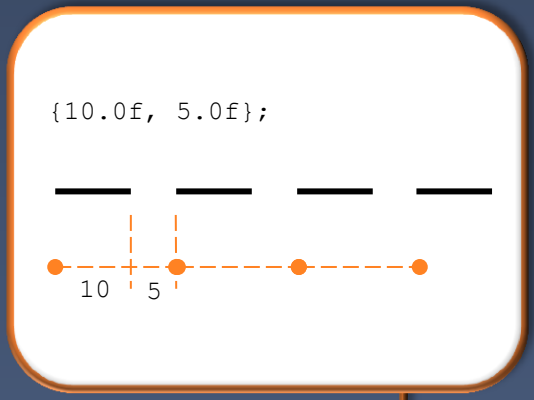
- *Establece el estilo del trazo y se aplica al contorno de una forma*
  - *Grosor*
  - *Estilo final de líneas*
  - *Estilo de unión de líneas*
  - *Discontinuidad*



BasicStroke()
BasicStroke(float width)
BasicStroke(float width, int cap, int join)
BasicStroke(float width, int cap, int join, float miterlimit)
BasicStroke(float width, int cap, int join, float miterlimit, float[] dash, float dash_phase)

Trazo (Stroke)

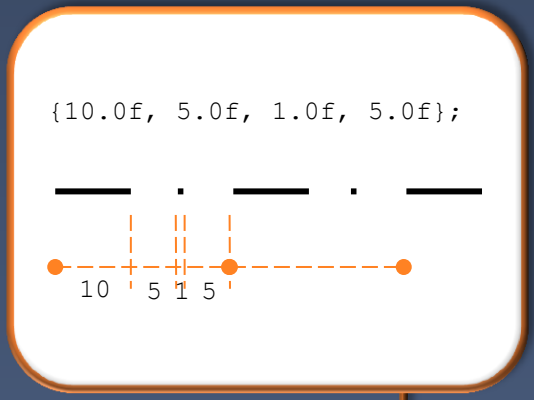
- *Establece el estilo del trazo y se aplica al contorno de una forma*
  - *Grosor*
  - *Estilo final de líneas*
  - *Estilo de unión de líneas*
  - *Discontinuidad*



BasicStroke()
BasicStroke(float width)
BasicStroke(float width, int cap, int join)
BasicStroke(float width, int cap, int join, float miterlimit)
BasicStroke(float width, int cap, int join, float miterlimit, float[] dash, float dash_phase)

Trazo (Stroke)

- *Establece el estilo del trazo y se aplica al contorno de una forma*
  - *Grosor*
  - *Estilo final de líneas*
  - *Estilo de unión de líneas*
  - *Discontinuidad*



BasicStroke()
BasicStroke(float width)
BasicStroke(float width, int cap, int join)
BasicStroke(float width, int cap, int join, float miterlimit)
BasicStroke(float width, int cap, int join, float miterlimit, float[] dash, float dash_phase)

Trazo (Stroke)

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    float patron[] = {10.0f, 10.0f};
    Stroke trazo = new BasicStroke(10.0f,BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND,0.0f
                                   patron,0.0f);
    g2d.setStroke(trazo);
    g2d.draw(linea);
    g2d.draw(rectangulo);
    .
    .
}
```

```
Line2D linea;
Rectangle2D rectangulo;
```

Contexto

TRAZO

RECORTE

RELLENO

FUENTE

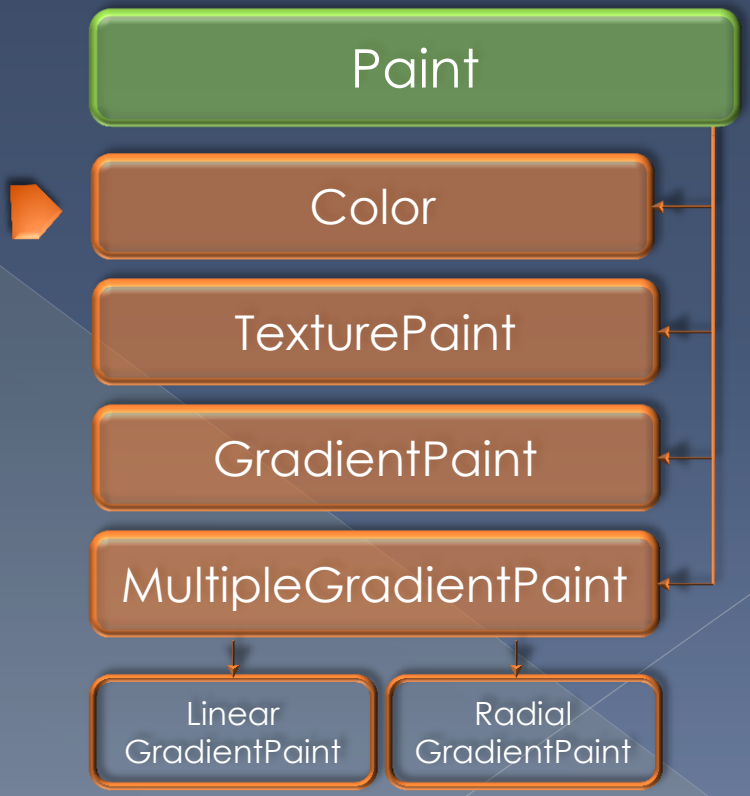
COMPOSICIÓN

RENDERING

TRANSFORMACIÓN

Relleno (Paint)

- *Establece el tipo y estilo de color del trazo y el relleno*
  - *Color liso*
  - *Basado en imagen*
  - *Color degradado*
    - *Lineal, 2 puntos*
    - *Lineal, N puntos*
    - *Radial*



Relleno (Paint)

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    Paint relleno = new Color(255,0,0);  // = Color.red
    g2d.setPaint(relleno);
    g2d.draw(linea);
    g2d.fill(rectangulo);
    .
    .
}
```

```
Line2D linea;
Rectangle2D rectangulo;
```

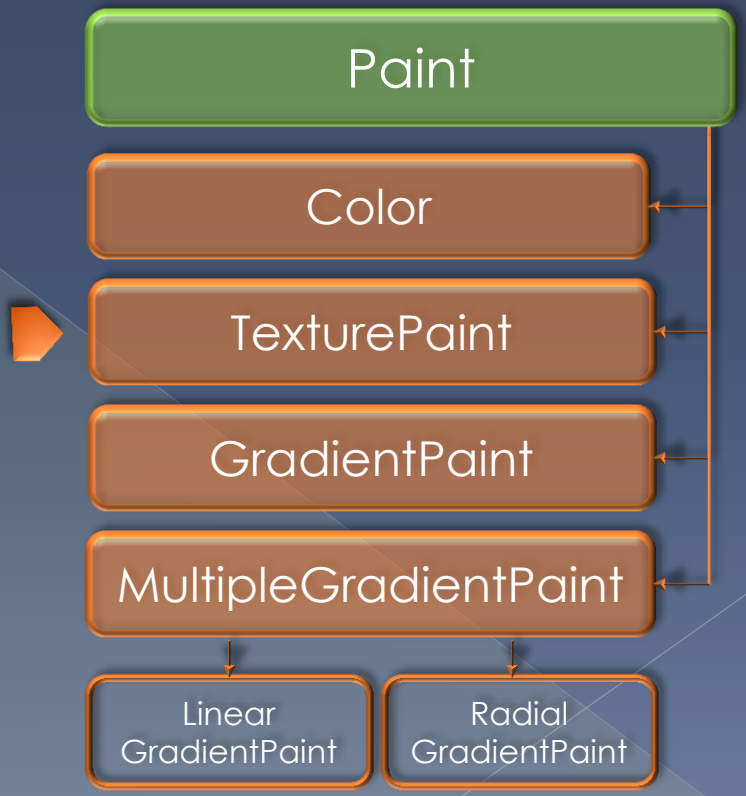
Relleno (Paint)

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;


    g2d.setPaint(Color.red);
    g2d.draw(linea);
    g2d.fill(rectangulo);
    .
    .
}
```

```
Line2D linea;
Rectangle2D rectangulo;
```

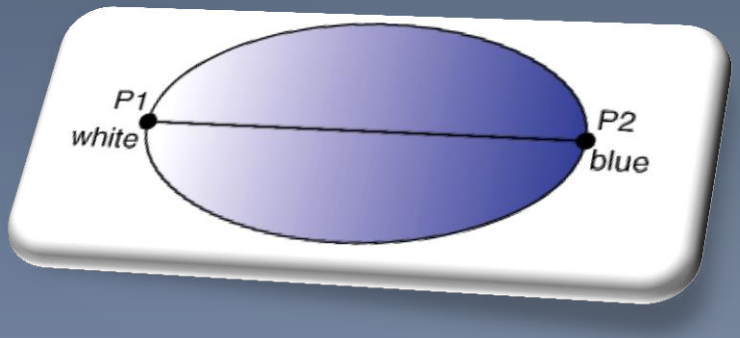
Relleno (Paint)

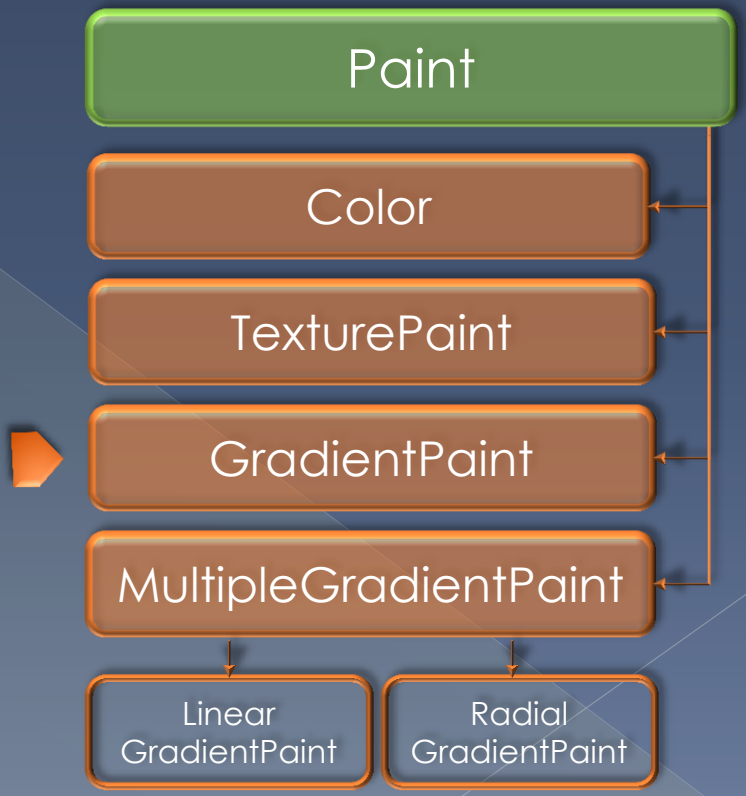
- *Establece el tipo y estilo de color del trazo y el relleno*
  - *Color liso*
  - *Basado en imagen*
  - *Color degradado*
    - *Lineal, 2 puntos*
    - *Lineal, N puntos*
    - *Radial*



Relleno (Paint)

- *Establece el tipo y estilo de color del trazo y el relleno*
  - *Color liso*
  - *Basado en imagen*
  - *Color degradado*
    - *Lineal, 2 puntos*
    - *Lineal, N puntos*
    - *Radial*

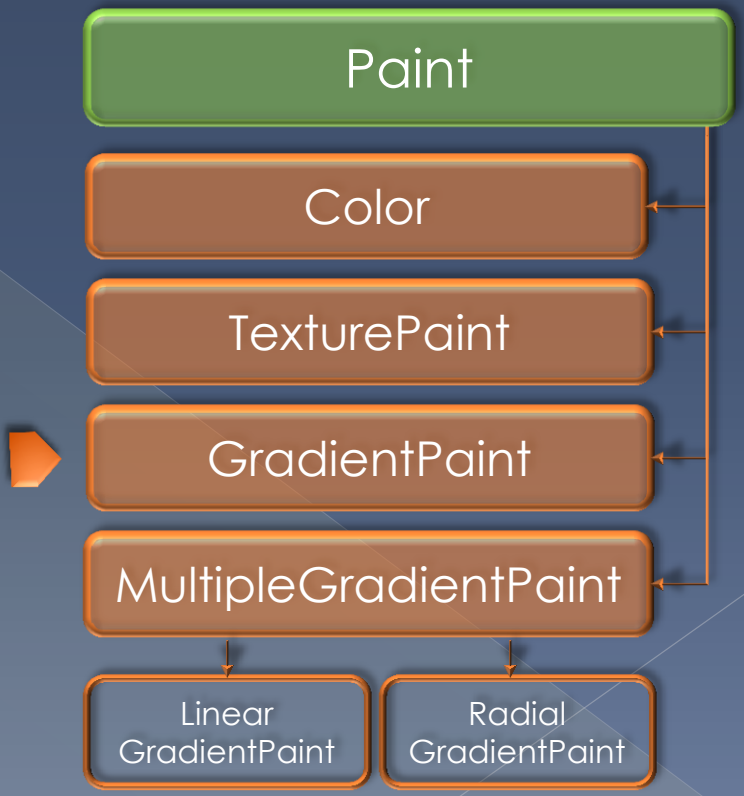




Relleno (Paint)

- *Establece el tipo y estilo de color del trazo y el relleno*
  - *Color liso*
  - *Basado en imagen*
  - *Color degradado*
    - *Lineal, 2 puntos*
    - *Lineal, N puntos*
    - *Radial*

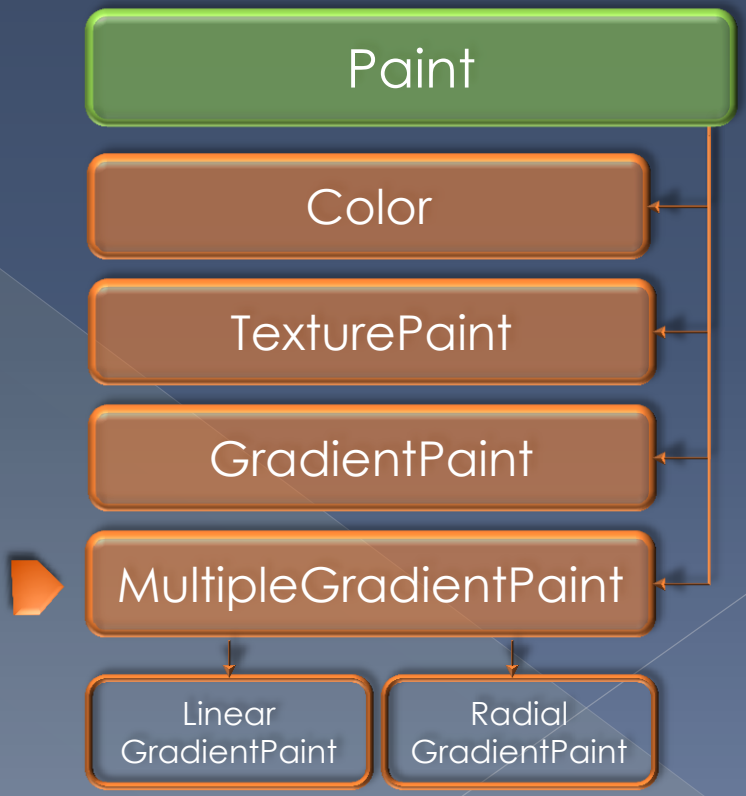
```
Point2D start = new Point2D.Float(0, 0);
Point2D end = new Point2D.Float(50, 50);
Color cs = Color.red, ce=Color.blue;

GradientPaint relleno =
                    new GradientPaint(start, cs, end, ce);
```



Relleno (Paint)

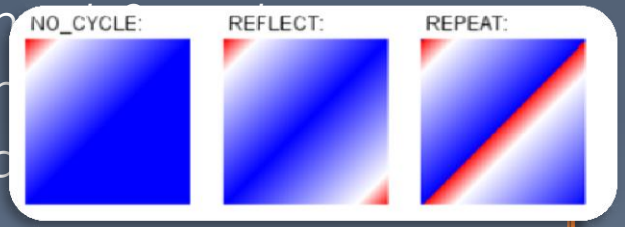
- *Establece el tipo y estilo de color del trazo y el relleno*
  - *Color liso*
  - *Basado en imagen*
  - *Color degradado*
    - *Lineal, 2 puntos*
    - *Lineal, N puntos*
    - *Radial*


```
Point2D start = new Point2D.Float(0, 0);
Point2D end = new Point2D.Float(50, 50);
float[] dist = {0.0f, 0.2f, 1.0f};
Color[] colors = {Color.RED, Color.WHITE, Color.BLUE};

LinearGradientPaint relleno =
        new LinearGradientPaint(start, end, dist, colors);
```

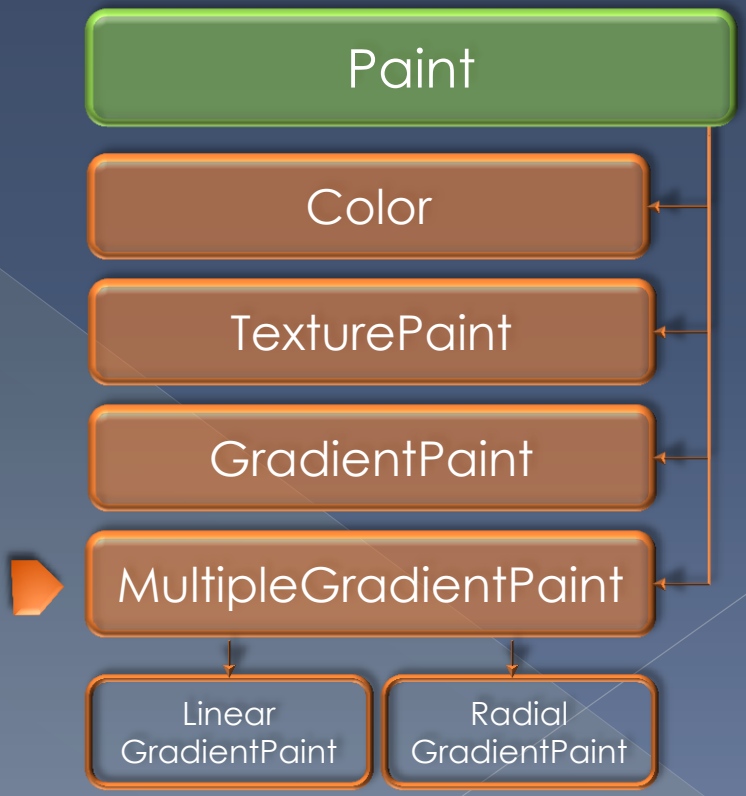


Relleno (Paint)

- *Establece el tipo y estilo de color del trazo y el relleno*
  - *Color liso*
  - *Basado en imagen*
  - *Color degradado*
    - *Lin*
    - *Lin*
    - *Ra*



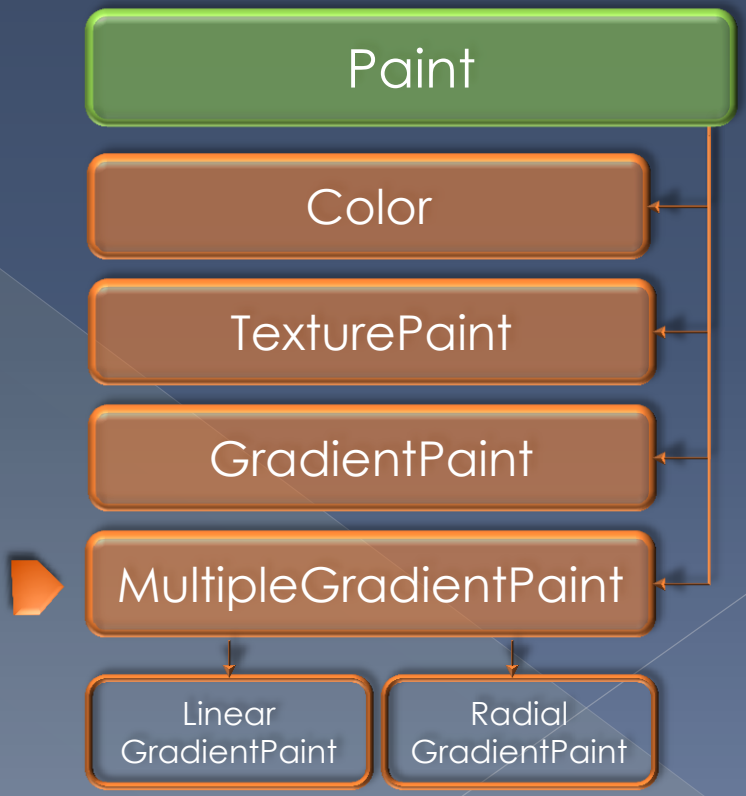

```
Point2D start = new Point2D.Float(0, 0);
Point2D end = new Point2D.Float(50, 50);
float[] dist = {0.0f, 0.2f, 1.0f};
Color[] colors = {Color.RED, Color.WHITE, Color.BLUE};

LinearGradientPaint relleno =
        new LinearGradientPaint(start, end, dist, colors);
```



Relleno (Paint)

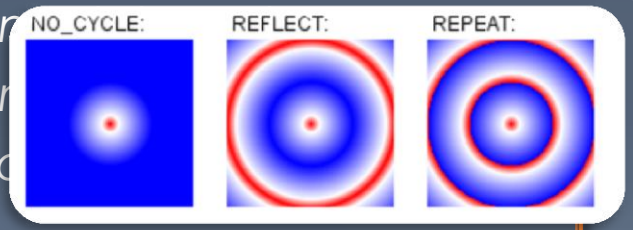
- *Establece el tipo y estilo de color del trazo y el relleno*
  - *Color liso*
  - *Basado en imagen*
  - *Color degradado*
    - *Lineal, 2 puntos*
    - *Lineal, N puntos*
    - *Radial*


```
Point2D center = new Point2D.Float(50, 50);
float r = 25;
float[] dist = {0.0f, 0.2f, 1.0f};
olor[] colors = {Color.RED, Color.WHITE, Color.BLUE};

RadialGradientPaint relleno =
        new RadialGradientPaint(center, r, dist, colors);
```

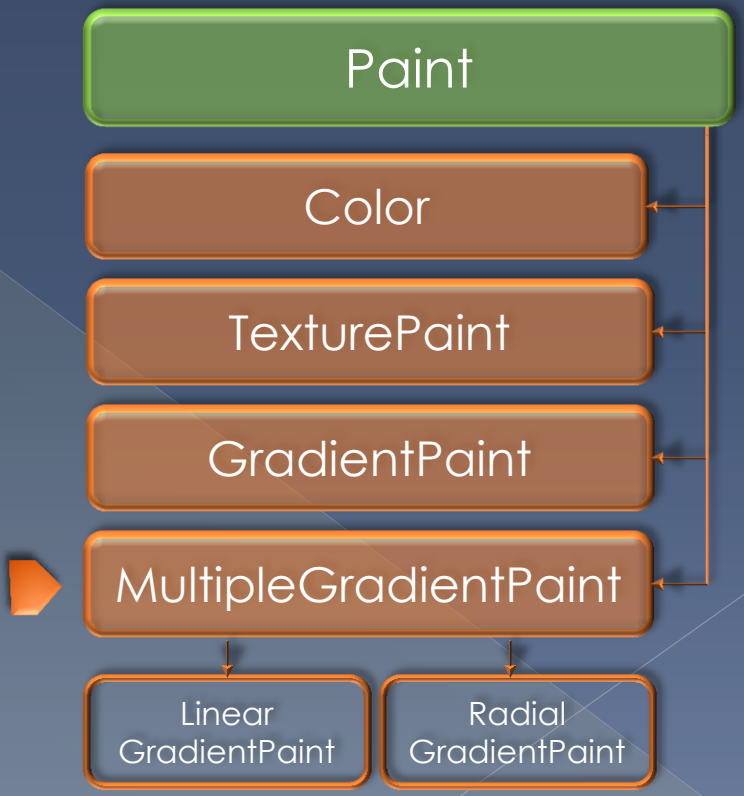


Relleno (Paint)

- *Establece el tipo y estilo de color del trazo y el relleno*
  - *Color liso*
  - *Basado en imagen*
  - *Color degradado*
    - *Lin*
    - *Lin*
    - *Ra*




```
Point2D center = new Point2D.Float(50, 50);
float r = 25;
float[] dist = {0.0f, 0.2f, 1.0f};
olor[] colors = {Color.RED, Color.WHITE, Color.BLUE};

RadialGradientPaint relleno =
        new RadialGradientPaint(center, r, dist, colors);
```



Contexto

TRAZO

RECORTE

RELLENO

FUENTE

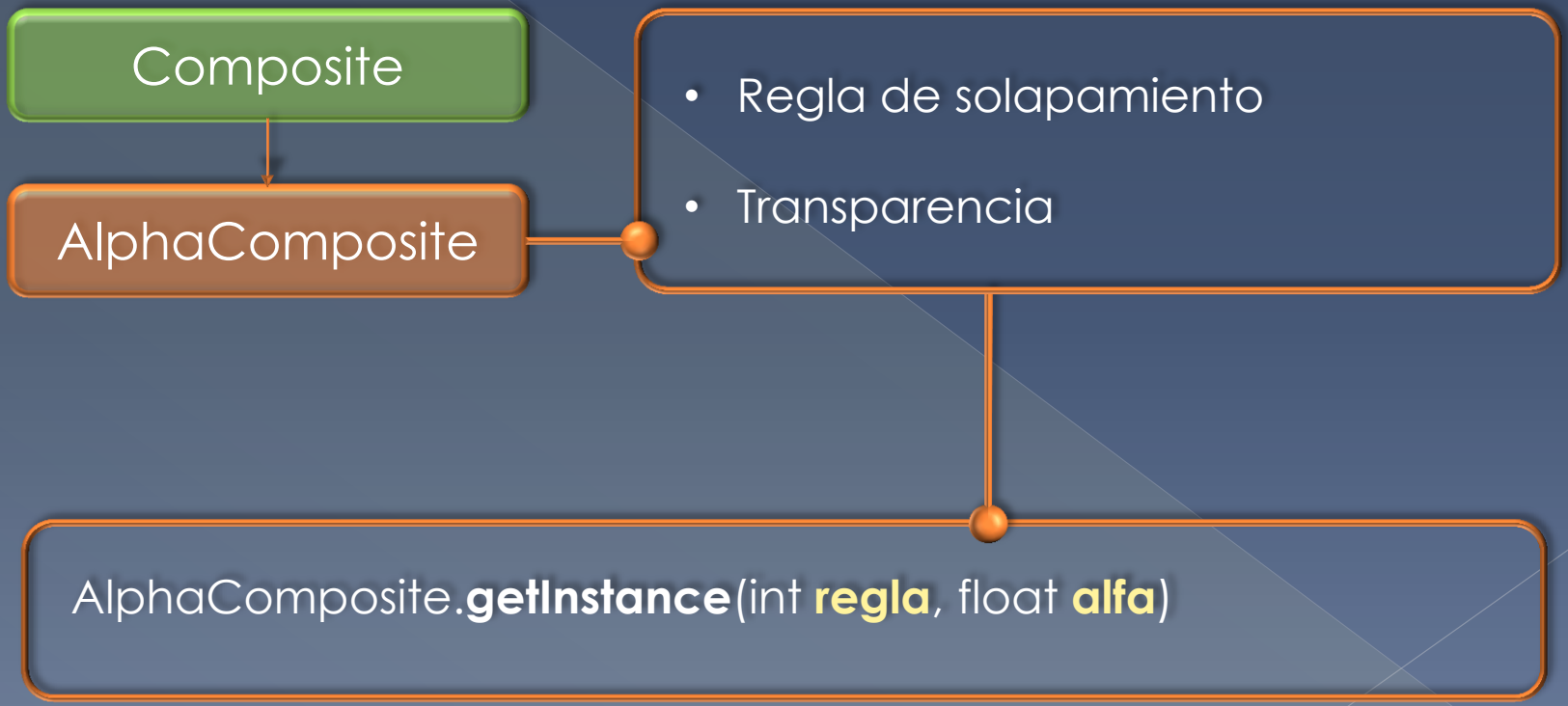
COMPOSICIÓN

RENDERING

TRANSFORMACIÓN

Composición

- *Establece la composición entre una nueva figura y el dibujo ya existente*



Composición

SRC_OVER

SRC_IN

SRC_OUT

CLEAR

DST_OVER

DST_IN

DST_OUT

Composición

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    Composite composicion = AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.5f);
    g2d.setComposite(composicion);
    g2d.fill(rectangulo);
    .
    .
}
```

```
Rectangle2D rectangulo;
```

Contexto

TRAZO

RECORTE

RELLENO

FUENTE

COMPOSICIÓN

RENDERING

TRANSFORMACIÓN

Transformación

- *Permite realizar transformaciones afines (rotaciones, traslaciones, escalados,…) en la visualización de una forma*
- *No modifica las coordenadas de la forma*

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & d_x & t_x \\ d_y & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$x' = s_x x + d_x y + t_x$$

$$y' = d_y x + s_y y + t_y$$

Transformación

- *Clase: AffineTransform*
- *Hay dos formas de construir un objeto:*
  - *Mediante el constructor, pasándole como parámetros la matriz de transformación*
  - *Mediante métodos estáticos, creando transformaciones predeterminadas*

AffineTransform.**getTranslateInstance(**double tx, double ty**)**

AffineTransform.**getScaleInstance(**double sx, double sy**)**

AffineTransform.**getRotateInstance(**double theta,**...)**

AffineTransform.**getShearInstance(**double shx, double shy**)**

Transformación

- *Concatenación de transformaciones*
  - *Mediante el método "concatenate"*

    concatenate(AffineTransform Tx)

  - *Mediante métodos específicos*

    translate(double tx, double ty**)**

    scale(double sx, double sy**)**

    rotate(double theta,**...)**

    shear(double shx, double shy**)**

Transformación

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    AffineTransform transformacion = AffineTransform.getScaleInstance(0.5,0.5);
    g2d.setTransform(transformacion);
    g2d.fill(r);
    .
    .
}
```

```
Rectangle2D r;
```

Transformación

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    AffineTransform transformacion = AffineTransform.getScaleInstance(0.5,0.5);
    transformacion.rotate(Math.toRadians(45.0));
    g2d.setTransform(transformacion);
    g2d.fill(r);
    .
    .
}
```

1. La concatenación hay que hacerla en el sentido "inverso" al efecto visual que se desee
2. Por defecto, el origen con respecto al cual se hacen las transformaciones en el (0,0)

```
Rectangle2D r;
```

Transformación

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    AffineTransform transformacion = new AffineTransform();
    transformacion.rotate(Math.toRadians(45.0),r.getCenterX(), r.getCenterY());
    g2d.setTransform(transformacion);
    g2d.fill(r);
    .
    .
}
```

1. La concatenación hay que hacerla en el sentido "inverso" al efecto visual que se desee
2. Por defecto, el origen con respecto al cual se hacen las transformaciones en el (0,0)

```
Rectangle2D r;
```

Transformación

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    AffineTransform transformacion = new AffineTransform();
    transformacion.translate(r.getCenterX(),r.getCenterY());
    transformacion.rotate(Math.toRadians(45.0);
    transformacion.translate(-r.getCenterX(),-r.getCenterY());
    g2d.setTransform(transformacion);
    g2d.fill(rect);
    .
    .
}
```

1. La concatenación hay que hacerla en el sentido "inverso" al efecto visual que se desee
2. Por defecto, el origen con respecto al cual se hacen las transformaciones en el (0,0)

```
Rectangle2D r;
```

Transformación

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    AffineTransform transformacion = new AffineTransform();
    transformacion.translate(r.getCenterX(),r.getCenterY());
    transformacion.scale(0.5,0.5);
    transformacion.rotate(Math.toRadians(45.0);
    transformacion.translate(-r.getCenterX(),-r.getCenterY());
    g2d.setTransform(transformacion);
    g2d.fill(rect);
    .
    .
}
```

1. La concatenación hay que hacerla en el sentido "inverso" al efecto visual que se desee
2. Por defecto, el origen con respecto al cual se hacen las transformaciones en el (0,0)

```
Rectangle2D r;
```

Contexto

TRAZO

RELLENO

COMPOSICIÓN

TRANSFORMACIÓN

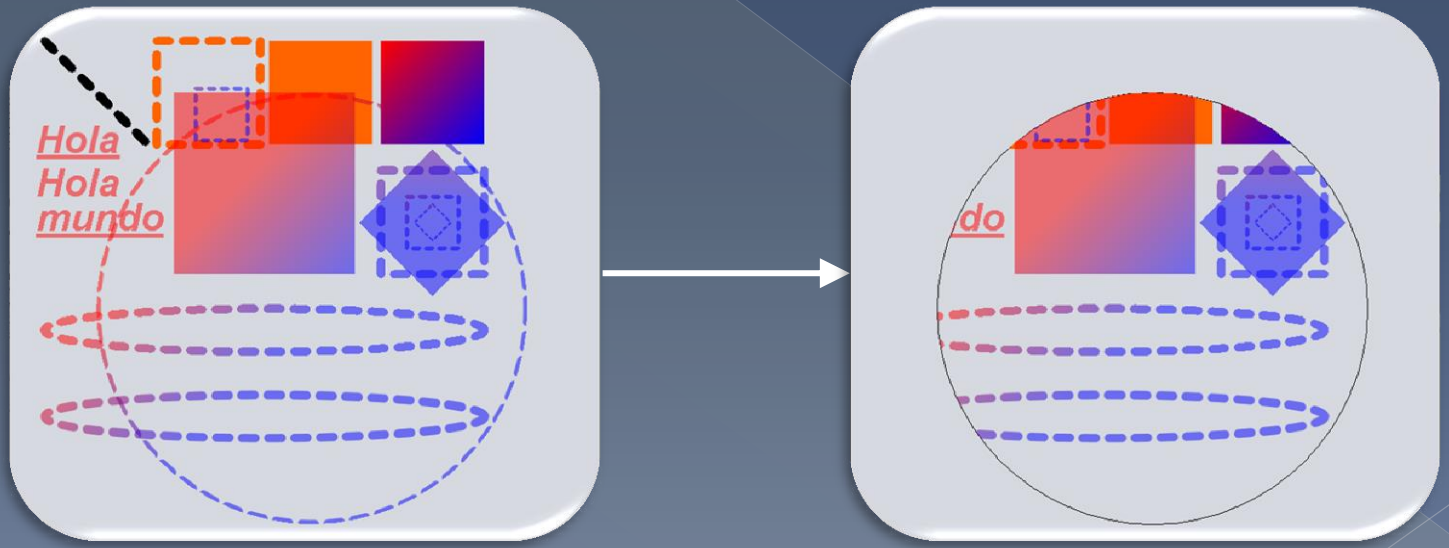
RECORTE

FUENTE

RENDERING

Recorte (Clip)

- *Define el área que será visualizada*
- *Para definir dicha área, se usará un Shape como parámetro*



Recorte (Clip)

```
public void  paint(Graphics g){
    super.paint(g);
    Graphics2D g2d = (Graphics2D)g;

    Shape areaRecorte = new Ellipse2D.Float(100,100,500,500);
    g2d.setClip(areaRecorte )
    g2d.fill(rectangulo);
    .
    .
}
```

```
Rectangle2D rectangulo;
```

Contexto

TRAZO

RELLENO

COMPOSICIÓN

TRANSFORMACIÓN

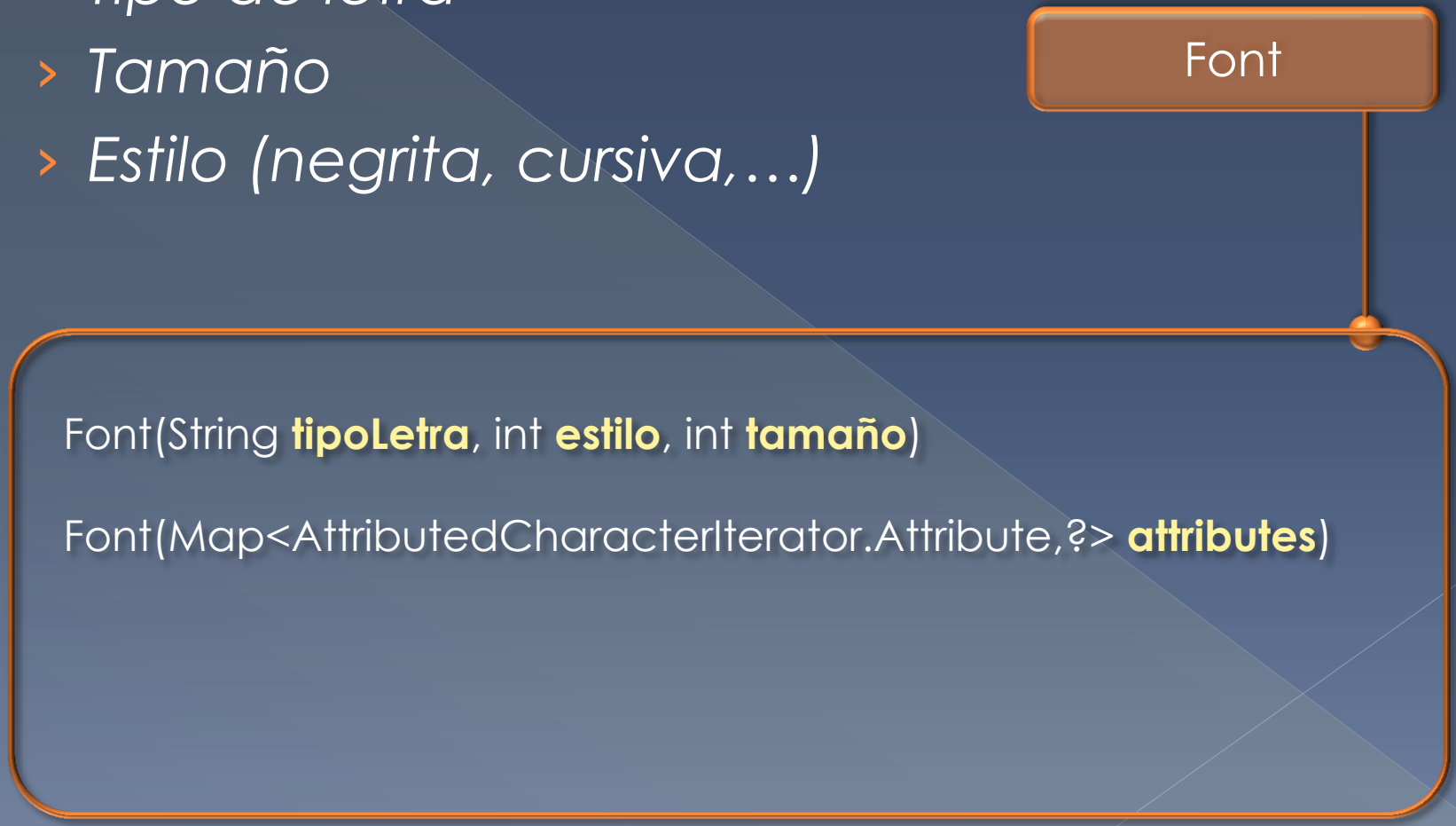
RECORTE

FUENTE

RENDERING

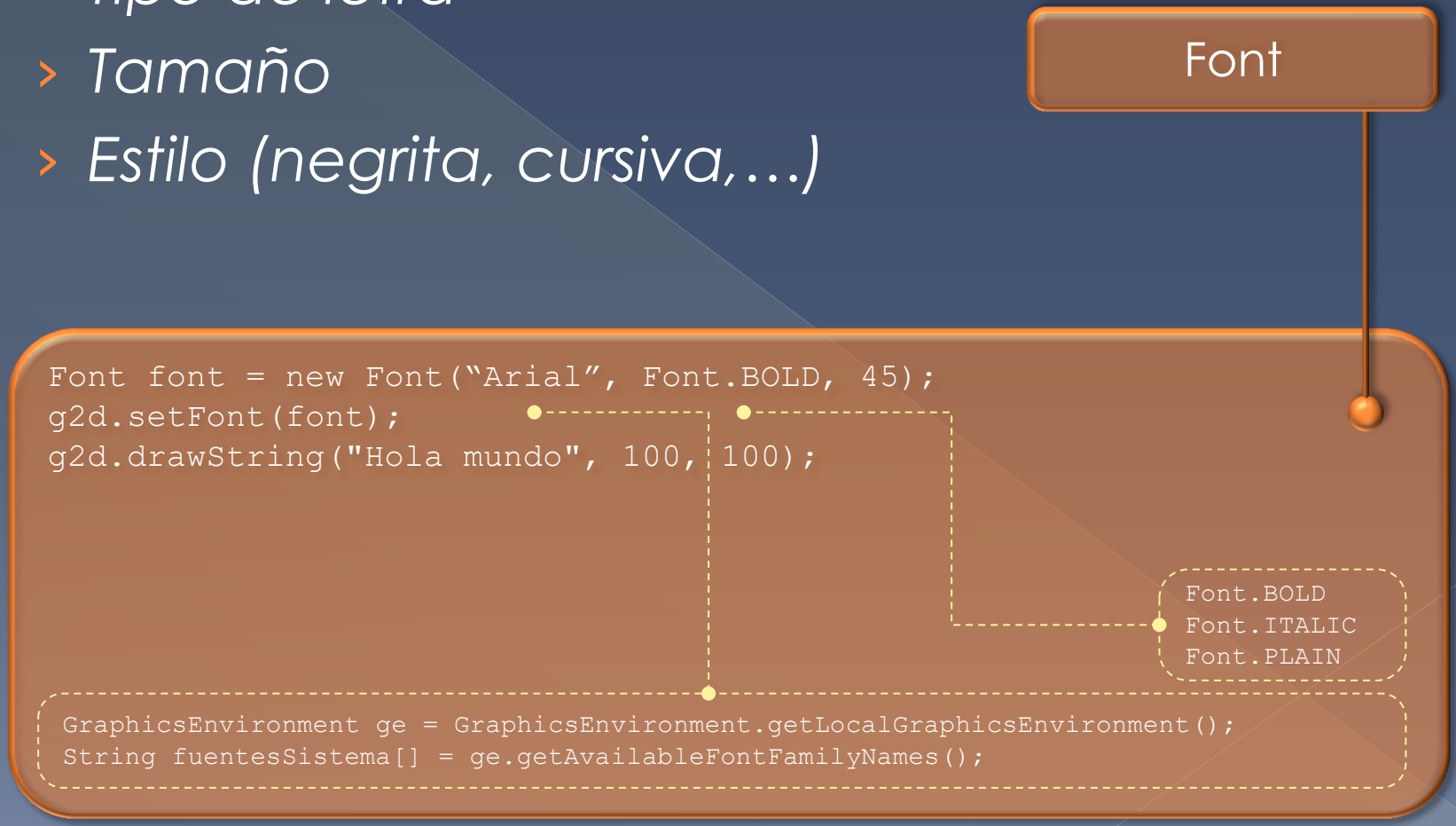
Fuente

- *Permite definir la fuente del texto*
  - *Tipo de letra*
  - *Tamaño*
  - *Estilo (negrita, cursiva,…)*



Fuente

- *Permite definir la fuente del texto*
  - *Tipo de letra*
  - *Tamaño*
  - *Estilo (negrita, cursiva,…)*



Fuente

- *Permite definir la fuente del texto*
  - *Tipo de letra*
  - *Tamaño*
  - *Estilo (negrita, cursiva,…)*

Fuente

- *La clase Font ofrece diferentes constantes de estilo (negrita, itálica), si bien éstas son muy limitadas. Para definir estilos más variados, se usa la clase TextAttribute*

```
Map atributos = new HashMap();
atributos.put(TextAttribute.UNDERLINE,TextAttribute.UNDERLINE_ON);
atributos.put(TextAttribute.SIZE, 40);
…
Font font = new Font(atributos);
g2d.setFont(font);
g2d.drawString("Hola mundo", 100, 100);
```

Fuente

- *La clase Font ofrece diferentes constantes de estilo (negrita, itálica), si bien éstas son muy limitadas. Para definir estilos más variados, se usa la clase TextAttribute*

```
Map atributos = otra_fuente.getAttributes();
atributos.put(TextAttribute.UNDERLINE,TextAttribute.UNDERLINE_ON);
atributos.put(TextAttribute.SIZE, 40);
…
Font font = new Font(atributos);
g2d.setFont(font);
g2d.drawString("Hola mundo", 100, 100);
```

Contexto

TRAZO

RECORTE

RELLENO

FUENTE

COMPOSICIÓN

RENDERING

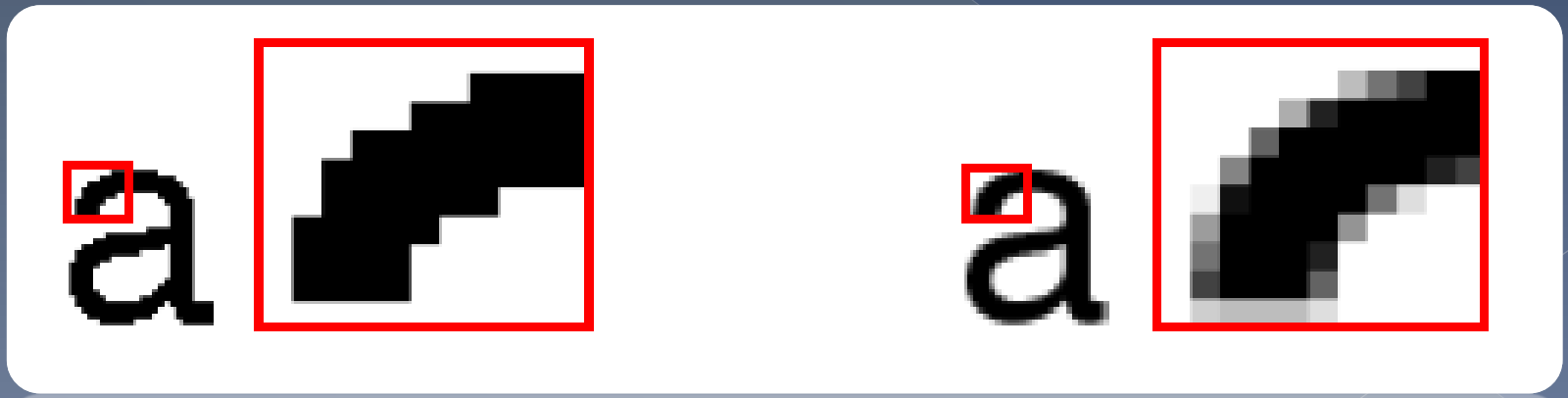
TRANSFORMACIÓN

RenderingHints

- *Permite mejorar la calidad del rendering*
- *Clase: RenderingHints*

RenderingHints(RenderingHints.Key **key**, Object **value**)

Propiedad a mejorar | Valor



RenderingHints

- *Permite mejorar la calidad del rendering*
- *Clase: RenderingHints*

RenderingHints(RenderingHints.Key **key**, Object **value**)

Propiedad a mejorar | Valor

```
public void paint (graphics g){
    Graphics2D g2 = (Graphics2D)g;
    RenderingHints rh = new RenderingHints(
                            RenderingHints.KEY_TEXT_ANTIALIASING,
                            RenderingHints.VALUE_TEXT_ANTIALIAS_ON);
    g2.setRenderingHints(rh);
    …
}
```

RenderingHints(Map<RenderingHints.Key,?> init)

RenderingHints

Hint	Key	Values
Antialiasing	KEY_ANTIALIASING	VALUE_ANTIALIAS_ON VALUE_ANTIALIAS_OFF VALUE_ANTIALIAS_DEFAULT
Alpha Interpolation	KEY_ALPHA_INTERPOLATION	VALUE_ALPHA_INTERPOLATION_QUALITY VALUE_ALPHA_INTERPOLATION_SPEED VALUE_ALPHA_INTERPOLATION_DEFAULT
Color Rendering	KEY_COLOR_RENDERING	VALUE_COLOR_RENDER_QUALITY VALUE_COLOR_RENDER_SPEED VALUE_COLOR_RENDER_DEFAULT
Dithering	KEY_DITHERING	VALUE_DITHER_DISABLE VALUE_DITHER_ENABLE VALUE_DITHER_DEFAULT
Fractional Text Metrics	KEY_FRACTIONALMETRICS	VALUE_FRACTIONALMETRICS_ON VALUE_FRACTIONALMETRICS_OFF VALUE_FRACTIONALMETRICS_DEFAULT
Image Interpolation	KEY_INTERPOLATION	VALUE_INTERPOLATION_BICUBIC VALUE_INTERPOLATION_BILINEAR VALUE_INTERPOLATION_NEAREST_NEIGHBOR
Rendering	KEY_RENDERING	VALUE_RENDER_QUALITY VALUE_RENDER_SPEED VALUE_RENDER_DEFAULT
Stroke Normalization Control	KEY_STROKE_CONTROL	VALUE_STROKE_NORMALIZE VALUE_STROKE_DEFAULT VALUE_STROKE_PURE
Text Antialiasing	KEY_TEXT_ANTIALIASING	VALUE_TEXT_ANTIALIAS_ON VALUE_TEXT_ANTIALIAS_OFF VALUE_TEXT_ANTIALIAS_DEFAULT VALUE_TEXT_ANTIALIAS_GASP VALUE_TEXT_ANTIALIAS_LCD_HRGB VALUE_TEXT_ANTIALIAS_LCD_HBGR VALUE_TEXT_ANTIALIAS_LCD_VRGB VALUE_TEXT_ANTIALIAS_LCD_VBGR