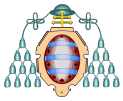


Práctica 4.
Pintado de componentes.

1 Método *paint*.

Se puede redefinir la apariencia gráfica de un componente redefiniendo el método *paint*. Este método es llamado cuando bien por el sistema bien por la aplicación y debería tener el código necesario para realizar el *pintado*.

NetBeans no proporciona un interfaz para redefinir el método *paint*, así que hay que hacerlo explícitamente. Para ello hay que crear una clase hija del componente que se quiera pintar y redefinir el método *paint*. Lo primero que tendrá hacer este método es llamar al método *paint* de la clase padre: `super.paint();`.

Es usual crear nuevos métodos para manejar datos del componente pero NetBeams crea el componente y se lo asigna a una referencia de la clase padre. En este caso es útil crear una referencia de la clase hija y asignarle el valor de la referencia creado por NetBeams utilizando casting.

Ejemplo: redefinir el método *paint* de un componente de clase *JPanel*.

Paso 1: Crear la clase *PaintPanel* que hereda de *JPanel*.

Paso 2: Crear el método *paint* como: `public void paint(Graphics g)`

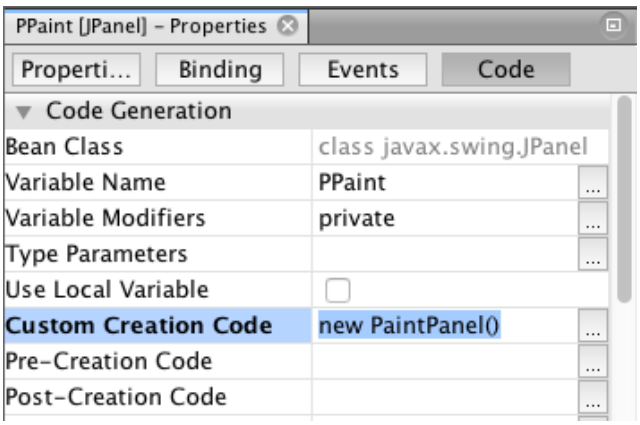
Paso 3: Llamar al método *paint* de la clase padre.

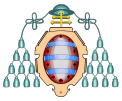
```
public class PaintPanel extends javax.swing.JPanel{

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

Paso 4: Sobre una *JFrame* con layout *GridLayout* crear un *JPanel*. Lo llamamos *PPaint*.

Paso 5: En la ventana de propiedades, sobre la pestaña *Code* modificar **Custom Creation Code** con: `new PaintPanel();`.





Paso 6: En ventana que use este panel crear una referencia *paintPanel* de la clase *PaintPanel*.

```
    private PaintPanel paintPanel; // Referencia al mismo objeto que PPaint
    // Variables declaration - do not modify
```

Paso 7: Asignar a la referencia *paintPanel* el valor de la referencia *PPaint* usando casting.

```
public class Paint extends javax.swing.JFrame {

    /** Creates new form PruCanvas ...3 lines */
    public Paint() {

        initComponents();
        this.setSize(700, 200);

        paintPanel=(PaintPanel)PPaint;
    }
```

1.1 Llamada desde el sistema.

El sistema siempre llama al método *paint* cuando se crea el componente y cuando necesita repintarlo, como por ejemplo cuando se cambia el tamaño del componente.

Para comprobar cuando es llamado se ha de incluir una traza que imprima un mensaje junto con un contador.

Ejemplo:

```
@Override
public void paint(Graphics g){
    super.paint(g);
    System.out.println("PaintPanel.paint:"+NPaint++);
    g.drawLine(0, 0, this.getWidth(), this.getHeight());
}
```

1.2 Llamada desde la aplicación.

Cuando en el programa cambie algo que implique que el pintado ha de cambiar también se ha de invocar el método *repaint()* que provoca una llamada asíncrona a *paint*.

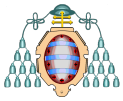
Para indicar las características de lo que se quiere pintar se utilizaran métodos *set* que modificarán campos del componente y métodos *get* que permiten obtener esos valores.

Ejemplo: permitir elegir el color de la línea pintada.

Paso 1: crear un menú **Formato** con un ítem **Color…**

Paso 2: en la clase *PaintPanel* crear el campo *lineColor* de clase *java.awt.Color* junto con los métodos *setLineColor* y *getLineColor* apropiados.

Paso 3: indicar en el método *paint* del panel que el color a usar es *lineColor*.



```
public void paint(Graphics g){
    super.paint(g);
    System.out.println("PaintPanel.paint:"+nPaint++);
    g.setColor(lineColor);
    g.drawLine(0, 0, this.getWidth(), this.getHeight());
}
```

Paso 4: ante la respuesta del evento asociado al ítem Color del menú, lanzar un diálogo de color que inicialmente muestre el color de la línea y que al aceptar un color cambie el color de la misma (usar la nueva referencia *paintPanel*). Forzar a repintar usando *repaint()*.

```
private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Color c=JColorChooser.showDialog(null, null,paintPanel.getLineColor());
    if(c!=null)
    {
        paintPanel.setLineColor(c);
        paintPanel.repaint();
    }
}
```

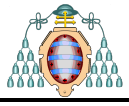
Actividad: Al hacer *clic* con el ratón sobre el panel se defina el punto de inicio de la recta pintada en ese punto. Utilizar los eventos *mouseClicked* y *mouseDragged*.

Para saber las coordenadas donde se ha hecho *clic*: `evt.getX()`, `evt.getY()`.

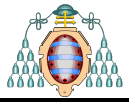
2 Clase *Graphics*.

Esta es la clase que permite realizar los pintados, se obtiene como parámetro del método *paint*. Tiene los métodos para pintar el componente. Son los siguientes:

Modifier and Type	Method and Description
abstract void	clearRect(int x, int y, int width, int height) Clears the specified rectangle by filling it with the background color of the current drawing surface.
abstract void	clipRect(int x, int y, int width, int height) Intersects the current clip with the specified rectangle.
abstract void	copyArea(int x, int y, int width, int height, int dx, int dy) Copies an area of the component by a distance specified by dx and dy.
abstract Graphics	create() Creates a new Graphics object that is a copy of this Graphics object.
Graphics	create(int x, int y, int width, int height) Creates a new Graphics object based on this Graphics object, but with a new translation and clip area.
abstract void	dispose() Disposes of this graphics context and releases any system resources that it is using.
void	draw3DRect(int x, int y, int width, int height, boolean raised) Draws a 3-D highlighted outline of the specified rectangle.
abstract void	drawArc(int x, int y, int width, int height, int startAngle, int arcAngle) Draws the outline of a circular or elliptical arc covering the specified rectangle.
void	drawBytes(byte[] data, int offset, int length, int x, int y) Draws the text given by the specified byte array, using this graphics context's current font and color.
void	drawChars(char[] data, int offset, int length, int x, int y) Draws the text given by the specified character array, using this graphics context's current font and color.



abstract boolean	drawImage(Image img, int x, int y, Color bgcolor, ImageObserver observer) Draws as much of the specified image as is currently available.
abstract boolean	drawImage(Image img, int x, int y, ImageObserver observer) Draws as much of the specified image as is currently available.
abstract boolean	drawImage(Image img, int x, int y, int width, int height, Color bgcolor, ImageObserver observer) Draws as much of the specified image as has already been scaled to fit inside the specified rectangle.
abstract boolean	drawImage(Image img, int x, int y, int width, int height, ImageObserver observer) Draws as much of the specified image as has already been scaled to fit inside the specified rectangle.
abstract boolean	drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, Color bgcolor, ImageObserver observer) Draws as much of the specified area of the specified image as is currently available, scaling it on the fly to fit inside the specified area of the destination drawable surface.
abstract boolean	drawImage(Image img, int dx1, int dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2, ImageObserver observer) Draws as much of the specified area of the specified image as is currently available, scaling it on the fly to fit inside the specified area of the destination drawable surface.
abstract void	drawLine(int x1, int y1, int x2, int y2) Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.
abstract void	drawOval(int x, int y, int width, int height) Draws the outline of an oval.
abstract void	drawPolygon(int[] xPoints, int[] yPoints, int nPoints) Draws a closed polygon defined by arrays of *x* and *y* coordinates.
void	drawPolygon(Polygon p) Draws the outline of a polygon defined by the specified Polygon object.
abstract void	drawPolyline(int[] xPoints, int[] yPoints, int nPoints) Draws a sequence of connected lines defined by arrays of *x* and *y* coordinates.
void	drawRect(int x, int y, int width, int height) Draws the outline of the specified rectangle.
abstract void	drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight) Draws an outlined round-cornered rectangle using this graphics context's current color.
abstract void	drawString(AttributedCharacterIterator iterator, int x, int y) Renders the text of the specified iterator applying its attributes in accordance with the specification of the TextAttribute class.
abstract void	drawString(String str, int x, int y) Draws the text given by the specified string, using this graphics context's current font and color.
void	fill3DRect(int x, int y, int width, int height, boolean raised) Paints a 3-D highlighted rectangle filled with the current color.
abstract void	fillArc(int x, int y, int width, int height, int startAngle, int arcAngle) Fills a circular or elliptical arc covering the specified rectangle.
abstract void	fillOval(int x, int y, int width, int height) Fills an oval bounded by the specified rectangle with the current color.
abstract void	fillPolygon(int[] xPoints, int[] yPoints, int nPoints) Fills a closed polygon defined by arrays of *x* and *y* coordinates.
void	fillPolygon(Polygon p) Fills the polygon defined by the specified Polygon object with the graphics context's current color.
abstract void	fillRect(int x, int y, int width, int height) Fills the specified rectangle.
abstract void	fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight) Fills the specified rounded corner rectangle with the current color.
void	finalize() Disposes of this graphics context once it is no longer referenced.
abstract Shape	getClip() Gets the current clipping area.
abstract	getClipBounds()



Rectangle	Returns the bounding rectangle of the current clipping area.
Rectangle	getClipBounds(Rectangle r) Returns the bounding rectangle of the current clipping area.
Rectangle	getClipRect() Deprecated. As of JDK version 1.1, replaced by getClipBounds().
abstract Color	getColor() Gets this graphics context's current color.
abstract Font	getFont() Gets the current font.
FontMetrics	getFontMetrics() Gets the font metrics of the current font.
abstract FontMetrics	getFontMetrics(Font f) Gets the font metrics for the specified font.
boolean	hitClip(int x, int y, int width, int height) Returns true if the specified rectangular area might intersect the current clipping area.
abstract void	setClip(int x, int y, int width, int height) Sets the current clip to the rectangle specified by the given coordinates.
abstract void	setClip(Shape clip) Sets the current clipping area to an arbitrary clip shape.
abstract void	setColor(Color c) Sets this graphics context's current color to the specified color.
abstract void	setFont(Font font) Sets this graphics context's font to the specified font.
abstract void	setPaintMode() Sets the paint mode of this graphics context to overwrite the destination with this graphics context's current color.
abstract void	setXORMode(Color c1) Sets the paint mode of this graphics context to alternate between this graphics context's current color and the new specified color.
String	toString() Returns a String object representing this Graphics object's value.
abstract void	translate(int x, int y) Translates the origin of the graphics context to the point (x, y) in the current coordinate system.