Dibujando

Graphics, Formas y Timers

La base

La ventana

- Esquema básico de una aplicación Swing.
  - Programa principal
  - VentanaBase que hereda de Jframe
  - Lámina que hereda de Jpanel

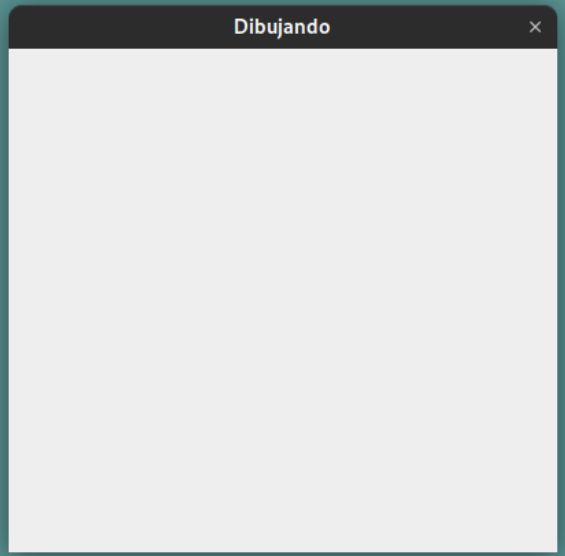
```java
public class Dibujando {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        VentanaBase ventana=new VentanaBase();
    }
}

2 usages
class VentanaBase extends JFrame{

    // Construimos una ventana con los atributos básicos
    1 usage
    public VentanaBase() {
        setVisible(true);
        setBounds( x: 400,  y: 200,  width: 400,  height: 400);
        setTitle("Dibujando");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        add(new Lamina());
    }
}

1 usage
class Lamina extends JPanel {

}
```


Dibujando

El Graphics (2D) y el paintComponent

PaintComponent(Graphics g)

- Las clases que heredan de JComponent tiene un método, paintComponent que se llama cada vez que se quiere mostrar ese componente en la pantalla.
- Podemos sobreescribir este método para dibujar o pintar en dicho componente.
- Vamos a pintar sobre la lámina, la clase que hereda de JPanel, sobreescribimos el método, y siempre llamamos al método padre y vamos a convertir el parámetro Graphics en un Graphics2D. A partir de aquí dibujaremos.

- Son muchos métodos que probar, los tenemos en la API de Graphics y de Graphics2D

```
class Lamina extends JPanel {

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2=(Graphics2D)g;
    }
}
```

Formas (de Graphics)

Algunos de los métodos principales para dibujar formas son:

- **drawLine**(int x1, int y1, int x2, int y2): dibuja una línea recta que conecta los puntos `(x1, y1)` y `(x2, y2)`.
- **drawRect**(int x, int y, int width, int height): dibuja un rectángulo de ancho `width` y altura `height`, con la esquina superior izquierda en las coordenadas `(x, y)`.
- **fillRect**(int x, int y, int width, int height): dibuja un rectángulo relleno de ancho `width` y altura `height`, con la esquina superior izquierda en las coordenadas `(x, y)`.
- **drawOval**(int x, int y, int width, int height): dibuja un óvalo de ancho `width` y altura `height`, con la esquina superior izquierda en las coordenadas `(x, y)`.
- **fillOval**(int x, int y, int width, int height): dibuja un óvalo relleno de ancho `width` y altura `height`, con la esquina superior izquierda en las coordenadas `(x, y)`.
- **drawPolygon**(int[] xPoints, int[] yPoints, int nPoints): dibuja un polígono definido por los puntos `(xPoints[i], yPoints[i])`, para `i` entre `0` y `nPoints-1`.
- **fillPolygon**(int[] xPoints, int[] yPoints, int nPoints): dibuja un polígono relleno definido por los puntos `(xPoints[i], yPoints[i])`, para `i` entre `0` y `nPoints-1`.
- **drawString**(String str, int x, int y): dibuja el texto especificado en la posición (x, y).
- **drawImage**(Image img, int x, int y, ImageObserver observer): dibuja la imagen especificada en la posición (x, y).

Otros métodos

- setColor(Color c): Establece el color para el siguiente dibujo.
- setFont(Font font): Establece la fuente para el siguiente dibujo.
- translate(int x, int y): Translada la posición de dibujo en el sistema de coordenadas por (x, y).
- rotate(double theta): Rota el sistema de coordenadas actual por un ángulo theta en radianes.
- scale(double sx, double sy): Escala el sistema de coordenadas actual por un factor sx horizontalmente y un factor sy verticalmente.

Otras clases

- De otros paquetes podemos acceder a otras clases para dibujar formas, como puede ser Arc2D, Ellipse2D, Rectangle2D, RoundRectangle2D

Primer ejemplo

Formas básicas

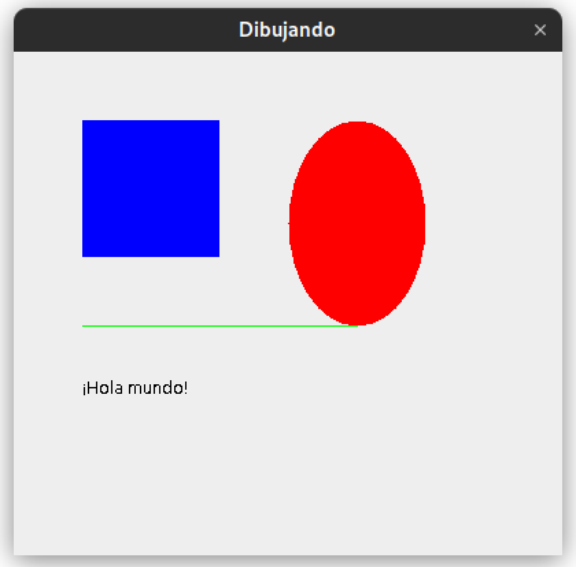
```java
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2=(Graphics2D)g;

    // Dibujar un cuadrado
    g2.setColor(Color.BLUE);
    g2.fillRect( i: 50,  i1: 50,  i2: 100,  i3: 100);

    // Dibujar un óvalo
    g2.setColor(Color.RED);
    g2.fillOval( i: 200,  i1: 50,  i2: 100,  i3: 150);

    // Dibujar una línea
    g2.setColor(Color.GREEN);
    g2.drawLine( i: 50,  i1: 200,  i2: 250,  i3: 200);

    // Dibujar una cadena de texto
    g2.setColor(Color.BLACK);
    g2.drawString( s: "¡Hola mundo!",  i: 50,  i1: 250);
}
```



Formas básicas
Usando java.awt.geom

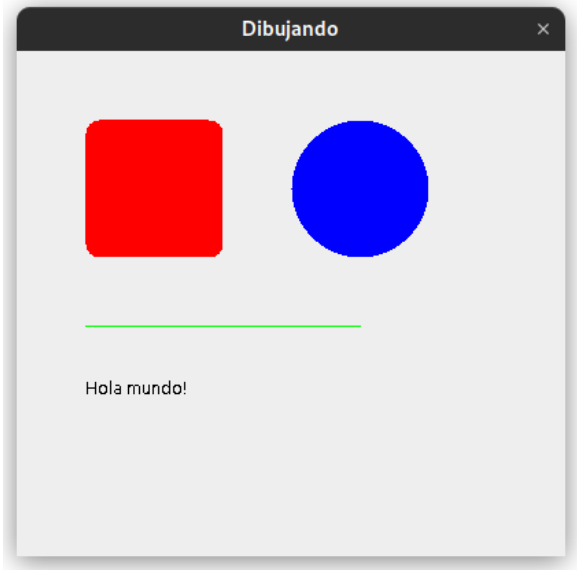
```java
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    Graphics2D g2d=(Graphics2D)g;

    // Dibujar un rectángulo redondeado
    g2d.setColor(Color.RED);
    RoundRectangle2D rect = new RoundRectangle2D.Double( x: 50,  y: 50,  w: 100,  h: 100,  arcw: 20,  arch: 20);
    g2d.fill(rect);

    // Dibujar un óvalo
    g2d.setColor(Color.BLUE);
    Ellipse2D ovalo = new Ellipse2D.Double( x: 200,  y: 50,  w: 100,  h: 100);
    g2d.fill(ovalo);

    // Dibujar una línea
    g2d.setColor(Color.GREEN);
    Line2D linea = new Line2D.Double( x1: 50,  y1: 200,  x2: 250,  y2: 200);
    g2d.draw(linea);

    // Dibujar una cadena de texto
    g2d.setColor(Color.BLACK);
    g2d.drawString( s: "Hola mundo!",  i: 50,  i1: 250);
}
```

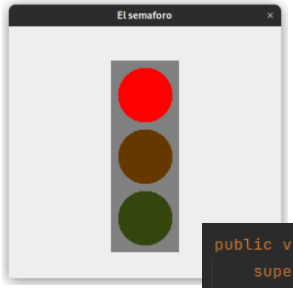


Ejercicio 1

Intenta realizar el siguiente dibujo.
Cambia colores y pon tu nombre en el título de la ventana

Animaciones

El semáforo



Seguimos con la misma estructura vista hasta ahora

```
public class Semaforo {
    public static void main(String[] args) {
        MarcoSemaforo ventana=new MarcoSemaforo();
    }
}
```

```
class MarcoSemaforo extends JFrame{

    1 usage
    public MarcoSemaforo() {
        setVisible(true);
        setBounds( x: 400,  y: 200,  width: 400,  height: 400);
        setTitle("El semaforo");
        setBackground(new Color( r: 81, g: 209, b: 246));
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        LaminaSemaforo semaforo=new LaminaSemaforo();
        add(semaforo);
    }
}
```

LaminaSemaforo hereda de JPanel y se pinta así

```
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    // Colores apagados
    Color rojo=new Color( r: 102,  g: 0,  b: 0);
    Color ambar=new Color( r: 102,  g: 56,  b: 0);
    Color verde=new Color( r: 52,  g: 69,  b: 13);
    Graphics2D g2=(Graphics2D)g;

    // Marco dle semáforo
    g2.setColor(Color.GRAY);
    Rectangle2D marco=new Rectangle2D.Double( x: 150, y: 50, w: 100, h: 280);
    g2.fill(marco);
    // 'Encendemos' el color seleccionado
    if (color==ColoresSemaforo.ROJO)
        rojo=Color.RED;
    else if (color==ColoresSemaforo.VERDE)
        verde=Color.GREEN;
    else if (color==ColoresSemaforo.AMBAR)
        ambar=Color.ORANGE;
    // Pintamos los círculos con los colores asignados
    g2.setColor(rojo);
    g2.fill(new Ellipse2D.Double( x: 160, y: 60, w: 80, h: 80));
    g2.setColor(ambar);
    g2.fill(new Ellipse2D.Double( x: 160, y: 150, w: 80, h: 80));
    g2.setColor(verde);
    g2.fill(new Ellipse2D.Double( x: 160, y: 240, w: 80, h: 80));
}
```

Clase Timer

Para realizar las animaciones lo que vamos a realizar es ejecutar un método cada cierto tiempo, y en la ejecución del método modificamos la posición de un elemento y repintamos, así da la impresión de movimiento.
En el semáforo lo que vamos a hacer es que cada cierto tiempo el color del semáforo 'encendido' va a cambiar.

La clase Timer se utiliza para programar tareas que se deben realizar después de un cierto retraso o en intervalos de tiempo regulares. Se puede usar para crear animaciones, temporizadores, relojes, juegos y otras aplicaciones que requieren un temporizador.

Para usar la clase se crea un objeto de la clase Timer, donde se especifica cada cuantos milisegundos se ejecutará y como segundo parámetro un objeto de una clase que implemente ActionListener, ya que se ejecutará el actionPerformed.

Encendiendo el semáforo

Hacemos que el panel implente el interfaz, ya que es ahí donde se va a 'mover'
class LaminaSemaforo extends JPanel implements ActionListener

Cada cierto tiempo se ejecutará el método. Y cambia el color encendido. Se llama al método para que se ejecute otra vez el método paintComponent

```
@Override
public void actionPerformed(ActionEvent arg0) {
    // TODO Auto-generated method stub
    if (color==ColoresSemaforo.ROJO)
        color=ColoresSemaforo.VERDE;
    else if (color==ColoresSemaforo.VERDE)
        color=ColoresSemaforo.AMBAR;
    else if (color==ColoresSemaforo.AMBAR)
        color=ColoresSemaforo.ROJO;
    repaint();
}
```

Se añaden 2 líneas a la ventana. Se crea el Timer, con el objeto que implementa el interfaz. Y la segunda inicia el temporizador

```
public MarcoSemaforo() {
    setVisible(true);
    setBounds( x: 400,  y: 200,  width: 400,  height: 400);
    setTitle("El semaforo");
    setBackground(new Color( r: 81, g: 209, b: 246));
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    LaminaSemaforo semaforo=new LaminaSemaforo();
    add(semaforo);
    Timer t=new Timer( delay: 1000,semaforo);
    t.start();
}
```

Ejercicio 2

Intenta añadir algo al dibujo y haz que se mueva por la pantalla, o que se mueva algo existente.

Pista. Lo que queramos modificar debe ser atributo de la clase para que sea accesible desde los métodos.

Enlace a video

Eventos