Solución ejercicio clase anterior

```
public void gradiente(int[][][] imagen)
{
     int celdasPorColor = 0;
     int avance = 1;

     celdasPorColor =  (int)Math.ceil((double)this.width/256.0);

     if(this.width<256.0)
          avance = (256)/this.width;
     else
          avance = Math.ceil(this.width/256.0);


     int color = 0;
     int opacidad = 255;

     for(int w=0; w<this.width; w+=celdasPorColor)
     {
          for(int i=0; i<celdasPorColor; i++)
          {
               for(int h=0; h<this.height;h++)
               {
                    imagen[w+i][h][1] = color;
                    imagen[w+i][h][3] = opacidad;
               }
          }

          opacidad-=avance;
          color+=avance;
     }

}
```

Solución ejercicio clase anterior

```
public void gradiente(int[][][] imagen)
{
     int celdasPorColor = 0;
     int avance = 1;

     celdasPorColor =  (int)Math.ceil((double)this.width/256.0);

     if(this.width<256.0)
          avance = (256)/this.width;
     else
          avance = Math.ceil(this.width/256.0);


     int color = 0;
     int opacidad = 255;

     for(int w=0; w<this.width; w+=celdasPorColor)
     {
          for(int i=0; i<celdasPorColor; i++)
          {
               for(int h=0; h<this.height;h++)
               {
                    imagen[w+i][h][1] = color;
                    imagen[w+i][h][3] = opacidad;
               }
          }

          opacidad-=avance;
          color+=avance;
     }

}
```

Solución ejercicio clase anterior

```
public void gradiente(int[][][] imagen)
{
     int celdasPorColor = 0;
     int avance = 1;

     celdasPorColor =  (int)Math.ceil((double)this.width/256.0);

     if(this.width<256.0)
          avance = (256)/this.width;
     else
          avance = Math.ceil(this.width/256.0);

     int color = 0;
     int opacidad = 255;

     for(int w=0; w<this.width; w+=celdasPorColor)
     {
          for(int i=0; i<celdasPorColor; i++)
          {
               for(int h=0; h<this.height;h++)
               {
                    imagen[w+i][h][1] = color;
                    imagen[w+i][h][3] = opacidad;
               }
          }

          opacidad-=avance;
          color+=avance;
     }

}
```

Solución ejercicio clase anterior

```
public void gradiente(int[][][] imagen)
{
     int celdasPorColor = 0;
     int avance = 1;

     celdasPorColor =  (int)Math.ceil((double)this.width/256.0);

     if(this.width<256.0)
          avance = (256)/this.width;
     else
          avance = Math.ceil(this.width/256.0);


     int color = 0;
     int opacidad = 255;

     for(int w=0; w<this.width; w+=celdasPorColor)
     {
          for(int i=0; i<celdasPorColor; i++)
          {
               for(int h=0; h<this.height;h++)
               {
                    imagen[w+i][h][1] = color;
                    imagen[w+i][h][3] = opacidad;
               }
          }

          opacidad-=avance;
          color+=avance;
     }

}
```

Solución ejercicio clase anterior

```
public void gradiente(int[][][] imagen)
{
     int celdasPorColor = 0;
     int avance = 1;

     celdasPorColor =  (int)Math.ceil((double)this.width/256.0);

     if(this.width<256.0)
          avance = (256)/this.width;
     else
          avance = Math.ceil(this.width/256.0);


     int color = 0;
     int opacidad = 255;

     for(int w=0; w<this.width; w+=celdasPorColor)
     {
          for(int i=0; i<celdasPorColor; i++)
          {
               for(int h=0; h<this.height;h++)
               {
                    imagen[w+i][h][1] = color;
                    imagen[w+i][h][3] = opacidad;
               }
          }

          opacidad-=avance;
          color+=avance;
     }

}
```

API Java 2d

API := Application programming interface

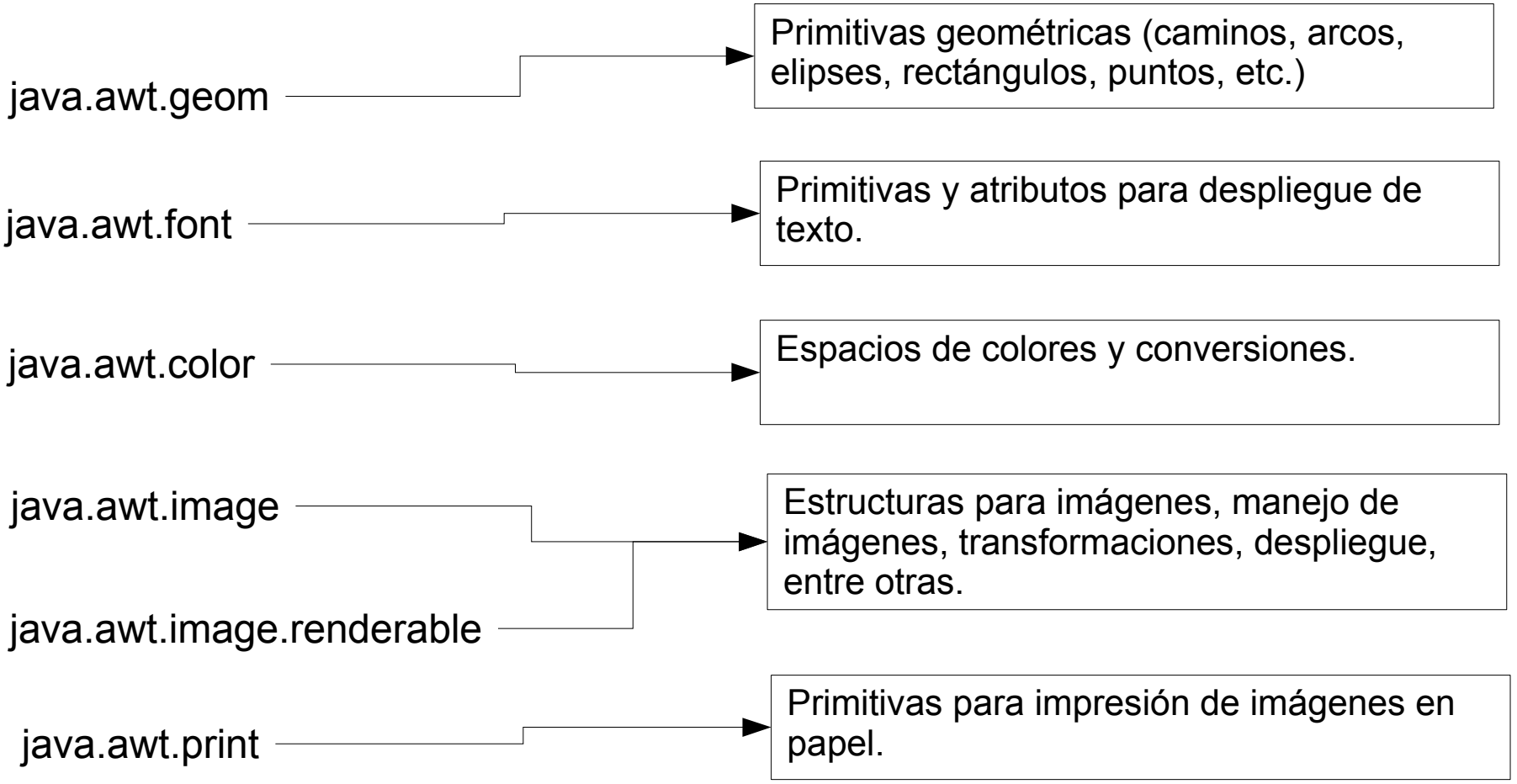
Es un conjunto de clases y sus funciones, empaquetadas. Pretenden otorgar herramientas de programación para facilitar un propósito particular.

API Java 2d :=

Centrada en aplicaciones gráficas bi-dimensionales: imágenes, dibujos, texto, animaciones. (Mantenida por Sun)

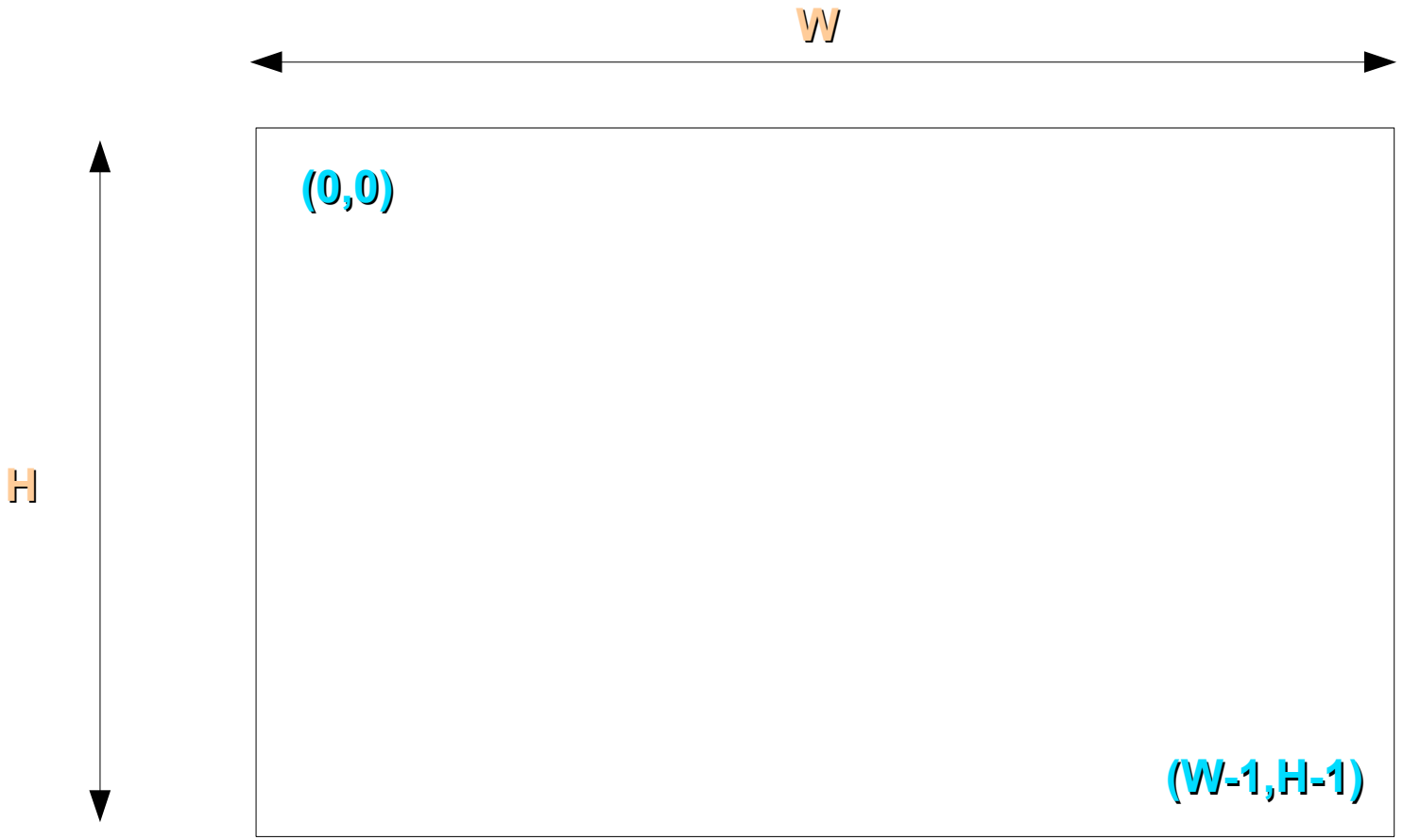
API Java 2d

Principales paquetes:



API Java 2d

Sistema de coordenadas



API Java 2d

Hola Mundo

API Java 2d

Hola mundo!

```
public class Ventana extends JFrame
{
     public static void main(String[] args)
     {

          /*   Crea este objeto y lo
          *    despliega
          */
     }

     public Ventana(int width, int height)
     {

          /*   Constructor, inicializa las variables de un objeto Jframe
          *    como el tamaño
          *    y los objetos que incluye
          */

     }

     public void paint(Graphics g)
     {
          /*   Todo objeto JFrame tiene un metodo como este que debe ser
          *    implementado. Este método se llama cada vez que el Jframe
          *    se mueve, maximiza, minimiza, cambia de aspecto, etc.
          */

     }

}
```

API Java 2d

Hola mundo!


```
public class Ventana extends JFrame
{
     public static void main(String[] args)
     {

          /*   Crea este objeto y lo
          *    despliega
          */
     }

     public Ventana(int width, int height)
     {

          /*   Constructor, inicializa las variables de un objeto Jframe
          *    como el tamaño
          *    y los objetos que incluye
          */

     }

     public void paint(Graphics g)
     {
          /*   Todo objeto JFrame tiene un metodo como este que debe ser
          *    implementado. Este método se llama cada vez que el Jframe
          *    se mueve, maximiza, minimiza, cambia de aspecto, etc.
          */

     }

}
```

API Java 2d

Hola mundo!


```
public class Ventana extends JFrame
{
	public static void main(String[] args)
	{

		/*	Crea este objeto y lo
		*	despliega
		*/
	}

	public Ventana(int width, int height)
	{

		/*	Constructor, inicializa las variables de un objeto Jframe
		*	como el tamaño
		*	y los objetos que incluye
		*/

	}

	public void paint(Graphics g)
	{
		/*	Todo objeto JFrame tiene un metodo como este que puede ser
		*	implementado. Este método se llama cada vez que el Jframe
		*	se mueve, maximiza, minimiza, cambia de aspecto, etc.
		*/

	}

}
```

API Java 2d

Hola mundo!

```
private static JFrame ventana;

public Ventana(int width, int height)
{

      ventana = this;

       ventana.setTitle("Nueva Ventana");

       ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JLabel emptyLabel = new JLabel("Dibujando...");

      emptyLabel.setPreferredSize(new Dimension(175, 100));


      Dimension telaDimensions = new Dimension(width,height);
      this.width = width;
      this.height = height;
      ventana.setSize(telaDimensions);



      ventana.pack();




}
```

API Java 2d

Hola mundo!

```
public void paint(Graphics g)
{

     Graphics2D graficos2D = (Graphics2D)g;
     graficos2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);



     Dimension dimensions = ventana.getSize();

     int gridWidth = (int)dimensions.getWidth();
     int gridHeight = (int)dimensions.getHeight();

     this.width = gridWidth;
     this.height = gridHeight;

     graficos2D.clearRect(0, 0, this.width, this.height);

     maxCharHeight = this.height/2;

     gridWidth = gridWidth - 100;

     FontMetrics fontMetrics = pickFont(graficos2D, "Hola Mundo", gridWidth);

     Color fg3D = Color.blue;

     graficos2D.setPaint(fg3D);

     graficos2D.drawString("Hola Mundo", 25, gridHeight/2);

     graficos2D.draw3DRect(5, 20, this.width - 20, this.height - 30, true);

}
```

API Java 2d

Hola mundo!

```
public void paint(Graphics g)
{

     Graphics2D graficos2D = (Graphics2D)g;
     graficos2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);



     Dimension dimensions = ventana.getSize();

     int gridWidth = (int)dimensions.getWidth();
     int gridHeight = (int)dimensions.getHeight();

     this.width = gridWidth;
     this.height = gridHeight;

     graficos2D.clearRect(0, 0, this.width, this.height);

     maxCharHeight = this.height/2;

     gridWidth = gridWidth - 100;

     FontMetrics fontMetrics = pickFont(graficos2D, "Hola Mundo", gridWidth);

     Color fg3D = Color.blue;

     graficos2D.setPaint(fg3D);

     graficos2D.drawString("Hola Mundo", 25, gridHeight/2);

     graficos2D.draw3DRect(5, 20, this.width - 20, this.height - 30, true);
}
```

API Java 2d

Formas v 1.0

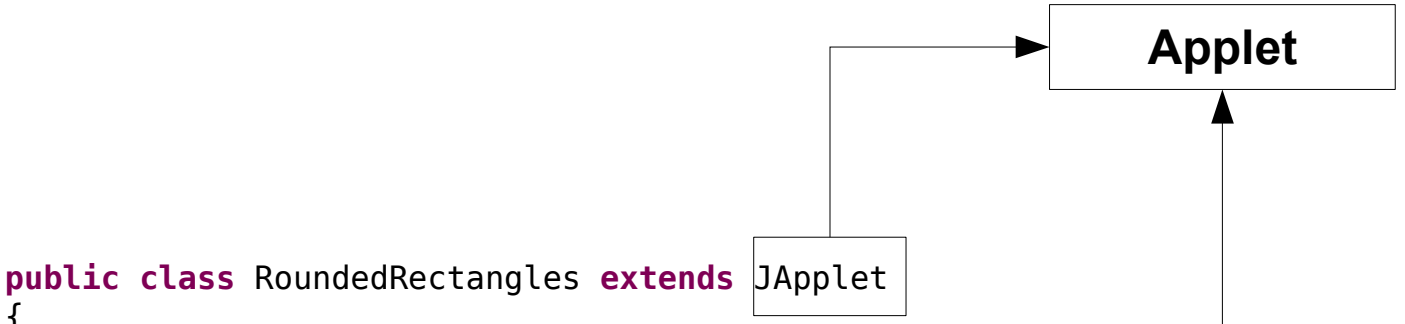
API Java 2d

Formas v 1.0

```
public class RoundedRectangles extends JApplet
{


    public void init()
    {
          /* Primer metodo que se llama al ejecutar
          *
          */

    }

    public void paint(Graphics g)
    {
          /* Método que se llama cada vez que se
          *  refresca la pantalla
          *
          */

    }



}
```

API Java 2d

Formas v 1.0




```
public class RoundedRectangles extends JApplet
{


    public void init()
    {
          /* Primer metodo que se llama al ejecutar
          *
          */

    }

    public void paint(Graphics g)
    {
          /* Método que se llama cada vez que se
          *  refresca la pantalla
          *
          */

    }



}
```

API Java 2d

Formas v 1.0

```
public void init()
{
    setBackground(Color.white);


}
```

```
public void paint(Graphics g)
    {
        Graphics2D graphics2D = (Graphics2D) g;

        graphics2D.clearRect(0, 0, width, height);

        graphics2D.setBackground(getBackground());

        graphics2D.clearRect(0, 0, width, height);

        graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);

        draw(width, height, graphics2D);


    }
```

API Java 2d

Formas v 1.0

```
  public void init()
  {
      setBackground(Color.white);


  }
```

```
public void paint(Graphics g)
    {
        Graphics2D graphics2D = (Graphics2D) g;
        graphics2D.clearRect(0, 0, width, height);
        graphics2D.setBackground(getBackground());
        graphics2D.clearRect(0, 0, width, height);
        graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
        draw(width, height, graphics2D);

    }
```

API Java 2d

Formas v 1.0

```
public void draw(int w, int h,  Graphics2D graphics2D)
    {

          Shape[] shapes = new Shape[shapeNumbers];
          double arcw = 0.5;
          double arch = 0.5;
          double step = 20.5;

          for(int i=0; i<shapeNumbers; i++)
          {
               int x1 = i*rectangleWidth;
               int y1 = i*rectangleWidth;
               int widthRectangle = w-2*rectangleWidth*i;
               int heightRectangle = h-2*rectangleWidth*i;
               shapes[i] = new RoundRectangle2D.Double(x1, y1, widthRectangle, heightRectangle, arcw, arch);
               graphics2D.setPaint(newFill(x1, y1, widthRectangle, heightRectangle));
               graphics2D.draw(shapes[i]);
               graphics2D.fill(shapes[i]);
          }



    }
```

API Java 2d

Formas v 1.0

```
public void draw(int w, int h,  Graphics2D graphics2D)
    {

        Shape[] shapes = new Shape[shapeNumbers];
        double arcw = 0.5;
        double arch = 0.5;
        double step = 20.5;

        for(int i=0; i<shapeNumbers; i++)
        {
             int x1 = i*rectangleWidth;
             int y1 = i*rectangleWidth;
             int widthRectangle = w-2*rectangleWidth*i;
        I    nt heightRectangle = h-2*rectangleWidth*i;

             shapes[i] = new RoundRectangle2D.Double(x1, y1, widthRectangle, heightRectangle, arcw, arch);
             graphics2D.draw(shapes[i]);

             graphics2D.setPaint(newFill(x1, y1, widthRectangle, heightRectangle));
             graphics2D.fill(shapes[i]);
        }


    }
```

API Java 2d

Formas v 1.0

```
private GradientPaint newFill(int x1, int y1, int x2, int y2)
{
     int r1 = (int)(Math.random()*255+1);
     int g1 = (int)(Math.random()*255+1);
     int b1 = (int)(Math.random()*255+1);
     int r2 = (int)(Math.random()*255+1);
     int g2 = (int)(Math.random()*255+1);
     int b2 = (int)(Math.random()*255+1);

     Color color1 = new Color(r1, g1, b1);
     Color color2 = new Color(r2, g2, b2);

     GradientPaint gradient = new GradientPaint(x1, y1, color1, x2, y2, color2, true);

     return gradient;

}
```

API Java 2d

Formas v 1.0

```
private GradientPaint newFill(int x1, int y1, int x2, int y2)
{
     int r1 = (int)(Math.random()*255+1);
     int g1 = (int)(Math.random()*255+1);
     int b1 = (int)(Math.random()*255+1);
     int r2 = (int)(Math.random()*255+1);
     int g2 = (int)(Math.random()*255+1);
     int b2 = (int)(Math.random()*255+1);

     Color color1 = new Color(r1, g1, b1);
     Color color2 = new Color(r2, g2, b2);

     GradientPaint gradient = new GradientPaint(x1, y1, color1, x2, y2, color2, true);

     return gradient;

}
```

API Java 2d

Formas v 2.0

API Java 2d

Formas v 2.0 (usando buffering)

```
public void paint(Graphics g)
{
     Graphics2D graphics2D = (Graphics2D) g;

     graphics2D.clearRect(0, 0, width, height);
     graphics2D.setBackground(getBackground());

     graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,RenderingHints.VALUE_ANTIALIAS_ON);


     graphics2D.drawImage(draw(width, height), 0, 0, this);


}
```

API Java 2d

Formas v 2.0 (usando buffering)

```
public void paint(Graphics g)
{
     Graphics2D graphics2D = (Graphics2D) g;

     graphics2D.clearRect(0, 0, width, height);
     graphics2D.setBackground(getBackground());

     graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING,RenderingHints.VALUE_ANTIALIAS_ON);

     graphics2D.drawImage(draw(width, height), 0, 0, this);

}
```

API Java 2d

Formas v 2.0 (usando buffering)

```
public BufferedImage  draw(int w, int h)
{

     BufferedImage buffer = (BufferedImage) createImage(w, h);
     Graphics2D graphics2D = buffer.createGraphics();

     Shape[] shapes = new Shape[shapeNumbers];
     double arcw = 0.5;
     double arch = 0.5;
     double step = 20.5;


     for(int i=0; i<shapeNumbers; i++)
     {
          int x1 = i*rectangleWidth;
          int y1 = i*rectangleWidth;
          int widthRectangle = w-2*rectangleWidth*i;
          int heightRectangle = h-2*rectangleWidth*i;

          shapes[i] = new RoundRectangle2D.Double(x1, y1, widthRectangle, heightRectangle, arcw, arch);
          graphics2D.setPaint(newFill(x1, y1, widthRectangle, heightRectangle));
          graphics2D.draw(shapes[i]);
          graphics2D.fill(shapes[i]);
     }

     return buffer;


}
```

API Java 2d

Formas v 2.0 (usando buffering)

```
public BufferedImage draw(int w, int h)
{
    BufferedImage buffer = (BufferedImage) createImage(w, h);
    Graphics2D graphics2D = buffer.createGraphics();

    Shape[] shapes = new Shape[shapeNumbers];
    double arcw = 0.5;
    double arch = 0.5;
    double step = 20.5;


    for(int i=0; i<shapeNumbers; i++)
    {
         int x1 = i*rectangleWidth;
         int y1 = i*rectangleWidth;
         int widthRectangle = w-2*rectangleWidth*i;
         int heightRectangle = h-2*rectangleWidth*i;

         shapes[i] = new RoundRectangle2D.Double(x1, y1, widthRectangle, heightRectangle, arcw, arch);
         graphics2D.setPaint(newFill(x1, y1, widthRectangle, heightRectangle));
         graphics2D.draw(shapes[i]);
         graphics2D.fill(shapes[i]);
    }
    return buffer;

}
```

API Java 2d

Animación

Animación con Doble Buffering

API Java 2d

Animación con Doble Buffering

```
public class Animation extends JApplet
{
    public void init() { // Lo mismo mas ahora Inicializa buffers }

    public void paint(Graphics g)
      {
            //Llama a draw(...)
            //Dibuja el buffer de lectura
            //Llama a repaint();
      }

    public void draw(int w, int h)
    {
            //Dibuja en el buffer de escritura
                  //Llama a Shape
                  //Llama a drawImage
            //Intercambia el buffer de lectura con el de escritura
    }

    private void drawImage(Graphics2D graphics2D, String filename)
    {
            //Dibuja en graphics2D una imagen leida desde un archivo
    }

    private void drawShape()
    {
            //Suma (circular con el borde) de la posicion
            //Setea nueva posicion en la figura
    }

    private void calculatePosition()
    {
            //Calcula nuevas posiciones (suma circular con el borde)
    }

    private GradientPaint newFill(int x1, int y1, int x2, int y2)
    {
            //Igual que antes, pero esta vez se llama sólo en init() [no queremos que cambien los colores]

    }
}
```

API Java 2d

Animación con Doble Buffering

```
public void init()
    {
        setBackground(Color.white);

        Dimension dimension = this.getSize();

        width = (int)dimension.getWidth();
        height = (int)dimension.getHeight();

        buffer1 = (BufferedImage) createImage(width, height);
        buffer2 = (BufferedImage) createImage(width, height);

        readBuffer = buffer1;
        writeBuffer = buffer2;


        ellipseWidth = (int)(Math.random()*width+25)/2;
        ellipseHeight = ellipseWidth;

        px = (int)(Math.random()*width+1);
        py = (int)(Math.random()*height+1);
        xStep = 1;
        yStep = 1;
        genericShape = new Ellipse2D.Double(px,py, ellipseWidth,ellipseHeight);

        gradient = newFill((int)px, (int)py, (int)ellipseWidth, (int)ellipseHeight);


    }
```

Animación con Doble Buffering

```
public void init()
    {
        setBackground(Color.white);

        Dimension dimension = this.getSize();

        width = (int)dimension.getWidth();
        height = (int)dimension.getHeight();

        buffer1 = (BufferedImage) createImage(width, height);
        buffer2 = (BufferedImage) createImage(width, height);

        readBuffer = buffer1;
        writeBuffer = buffer2;

        ellipseWidth = (int)(Math.random()*width+25)/2;
        ellipseHeight = ellipseWidth;

        px = (int)(Math.random()*width+1);
        py = (int)(Math.random()*height+1);
        xStep = 1;
        yStep = 1;
        genericShape = new Ellipse2D.Double(px,py, ellipseWidth,ellipseHeight);

        gradient = newFill((int)px, (int)py, (int)ellipseWidth, (int)ellipseHeight);

    }
```

API Java 2d

Animación con Doble Buffering

```
public void draw(int w, int h)
    {

        Graphics2D graphics2D = (Graphics2D)(writeBuffer.getGraphics());


        graphics2D.clearRect(0, 0, width, height);

        graphics2D.setBackground(getBackground());

        graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);

        drawImage(graphics2D, imageFile);

        drawShape();
        graphics2D.draw(genericShape);
        graphics2D.setPaint(gradient);
        graphics2D.fill(genericShape);

        BufferedImage tmpBuffer = readBuffer;

        readBuffer = writeBuffer;
        writeBuffer = tmpBuffer;



    }
```

API Java 2d

Animación con Doble Buffering

```
public void draw(int w, int h)
    {
        Graphics2D graphics2D = (Graphics2D)(writeBuffer.getGraphics());

        graphics2D.clearRect(0, 0, width, height);

        graphics2D.setBackground(getBackground());

        graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);

        drawImage(graphics2D, imageFile);

        drawShape();
        graphics2D.draw(genericShape);
        graphics2D.setPaint(gradient);
        graphics2D.fill(genericShape);

        BufferedImage tmpBuffer = readBuffer;

        readBuffer = writeBuffer;
        writeBuffer = tmpBuffer;


    }
```

API Java 2d

Animación con Doble Buffering

```
public void draw(int w, int h)
    {

        Graphics2D graphics2D = (Graphics2D)(writeBuffer.getGraphics());


        graphics2D.clearRect(0, 0, width, height);

        graphics2D.setBackground(getBackground());

        graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);

        drawImage(graphics2D, imageFile);

        drawShape();
        graphics2D.draw(genericShape);
        graphics2D.setPaint(gradient);
        graphics2D.fill(genericShape);

        BufferedImage tmpBuffer = readBuffer;

        readBuffer = writeBuffer;
        writeBuffer = tmpBuffer;



    }
```

API Java 2d

Animación con Doble Buffering

```
public void draw(int w, int h)
    {

        Graphics2D graphics2D = (Graphics2D)(writeBuffer.getGraphics());


        graphics2D.clearRect(0, 0, width, height);

        graphics2D.setBackground(getBackground());

        graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);

        drawImage(graphics2D, imageFile);

        drawShape();
        graphics2D.draw(genericShape);
        graphics2D.setPaint(gradient);
        graphics2D.fill(genericShape);

        BufferedImage tmpBuffer = readBuffer;

        readBuffer = writeBuffer;
        writeBuffer = tmpBuffer;



    }
```

API Java 2d

Animación con Doble Buffering

```
public void draw(int w, int h)
    {

        Graphics2D graphics2D = (Graphics2D)(writeBuffer.getGraphics());


        graphics2D.clearRect(0, 0, width, height);

        graphics2D.setBackground(getBackground());

        graphics2D.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);

        drawImage(graphics2D, imageFile);

        drawShape();
        graphics2D.draw(genericShape);
        graphics2D.setPaint(gradient);
        graphics2D.fill(genericShape);

        BufferedImage tmpBuffer = readBuffer;

        readBuffer = writeBuffer;
        writeBuffer = tmpBuffer;


    }
```

API Java 2d

Animación con Doble Buffering

```
public void paint(Graphics g)
{
     Graphics2D graphics2D = (Graphics2D)g;

     draw(width, height);

     graphics2D.drawImage(readBuffer, 0, 0, this);

     repaint();

}
```

API Java 2d

Animación con Doble Buffering

```
public void paint(Graphics g)
{
     Graphics2D graphics2D = (Graphics2D)g;


     draw(width, height);

     graphics2D.drawImage(readBuffer, 0, 0, this);

     repaint();

}

private void drawImage(Graphics2D graphics2D, String filename)
{
     Image image = getToolkit().getImage(filename);

     graphics2D.drawImage(image, 0, 0, this);
}

private void drawShape()
{
     calculatePosition();
     genericShape.setFrame(px,py, ellipseWidth, ellipseHeight);
}
```

API Java 2d

Animación con Doble Buffering

```
private void drawImage(Graphics2D graphics2D, String filename)
    {
     Image image = getToolkit().getImage(filename);

     graphics2D.drawImage(image, 0, 0, this);
    }

    private void drawShape()
    {
     calculatePosition();
     genericShape.setFrame(px,py, ellipseWidth, ellipseHeight);
    }

    private void calculatePosition()
    {
     px=px+xStep;
     py=py+yStep;

     if((px+ellipseWidth>=width)&&(xStep==1))
          xStep = -1;
     else if((px<0)&&(xStep==-1))
          xStep=1;

     if((py+ellipseHeight>=height)&&(yStep==1))
          yStep =-1;
     else if((py<0)&&(yStep==-1))
          yStep=1;
    }

    private GradientPaint newFill(int x1, int y1, int x2, int y2)
    {
     Color color1 = new Color((int)(Math.random()*255+1), (int)(Math.random()*255+1), (int)
(Math.random()*255+1));
     Color color2 = new Color((int)(Math.random()*255+1), (int)(Math.random()*255+1), (int)
(Math.random()*255+1));

     GradientPaint gradient = new GradientPaint(x1, y1, color1, x2, y2, color2, true);
     return gradient;

    }
```

Ejercicio

Implementar un JApplet que dibuje rectángulos en una posición aletatoria, de ancho y alto aleatorios y con un gradiente de colores aleatorios. En la pantalla aparece sólo un rectángulo a la vez, que cambia de propiedades (alto, ancho, posición y gradiente) cada 0.5 segundos. Debe implementar las funciones paint, draw y drawRectangle. Paint debe utilizar doble buffering, graficar el buffer de lectura y debe llamar a draw, draw debe dibujar en el buffer de escritura llamando a drawNewRectangle() y luego debe intercambiar los buffers, mientras que drawNewRect debe generar un nuevo rectángulo con nuevas propiedades.

Hint 1: Notar que las funciones paint y draw se mantienen, salvo pequeños cambios.
Hint 2: Considere que ya está implementada la función GradientPaint newFill() y que no necesita ingresarle parámetros.
Hint 3: Para dormir el proceso actual 0,5 segundos, puede utilizar la función Thread.sleep(500, 0)

```
public void paint(Graphics g)

public void draw(int w, int h)

private void drawNewRectangle()
```