.paquete javax.swing

.

***Para interfaces de usuario, para ventanas. Java trae una serie de clases, de metodos para poder crear estas ventanas.***

***Java nos proporciona el paquete ‘swing’ donde se ecuentran todas estas clases y métodos.***

***Anteriormente java tenía la clase de awt. Está bien para aplicaciones sencillas.***

***El paquete ‘swing’ ofrece una eficiencia mucho mayor. Permite mantener la apariencia de las ventanas en cualquier SO que se ejecute.***

***PARA CREAR UN FRAME O VENTANA, TENEMOSS QUE CREAR UNA CLASE QUE HEREDE DE LA CLASE Jframe, al invocar el constructor creamos un marco, y luego le aplicamos mt, para hacerle visible, dar tamaño etc.***

PRIMER VIDEO VIDEO 55

PARA CREAR BOTONES, VENTANAS etc.

Swing es un paquete creado por java para que las ***interfaces graficas,*** hechas por los programadores, tengan el mismo aspecto en todos los S.O, cuando se ejecuten.

Con el paquete **java.awt** las interfaces toman aspectos diferentes dependiendo del S.O en el que se ejeuten (*solaris, windows, etc).*

**DATO:**

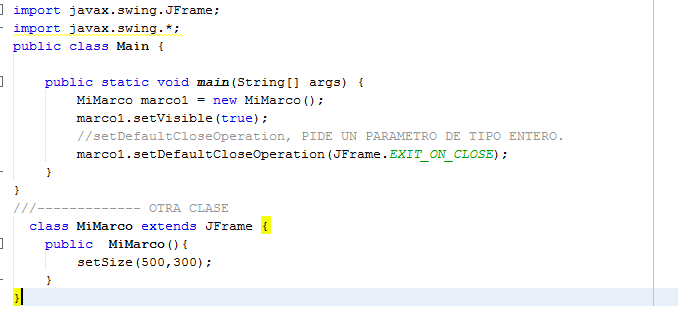
, marcos, ventanas y frames es lo mismo, son ventanas

Caracteristicas;

Son invisible, no tiene tamaño y es obligatorio indicar, a la ventana lo que queremos que haga cuando se cierre.

ventana.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

***CREAMOS UNA VENTANA.***

. 

. Son invisible ------------------------------------- marco1.setVisible(true);

, no tiene tamaño------------------------------- setSize(500,300);

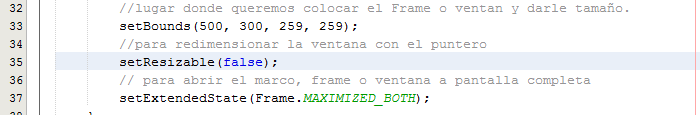
, haga cuando se cierre---------------------- marco1.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

2º VIDEO VIDEO 56

***COLOCAMOS LA VENTANA EN UNA DETERMINADA PARTE DE LA PANTALLA.***

***PONEMOS UN TITULO A LA VENTANA.***

Gracias a la gran cantidad de clases de las que puede heredar la clase Jframe, disponemos de muchos mt, para poder hacer las tres cosas.



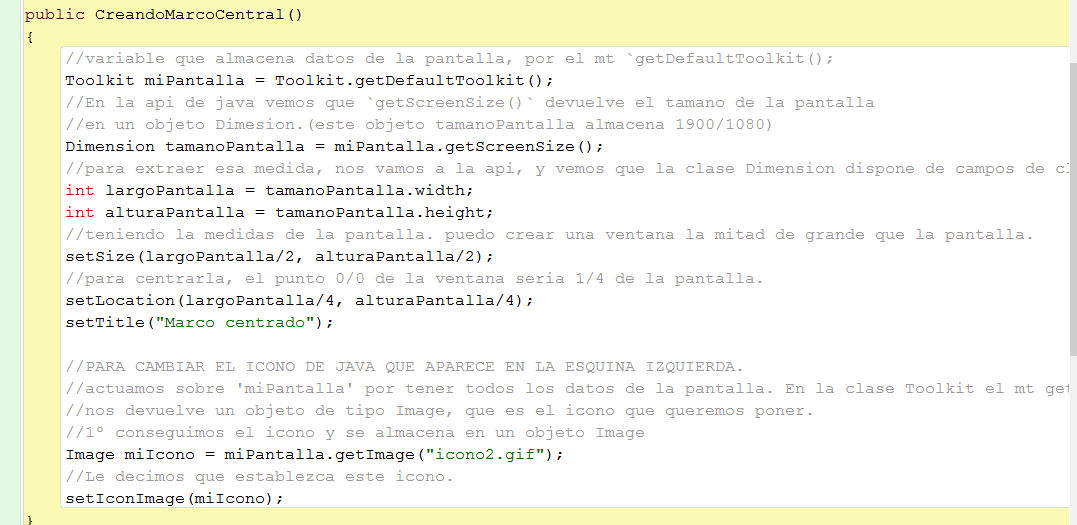
VIDEO VIDEO 57

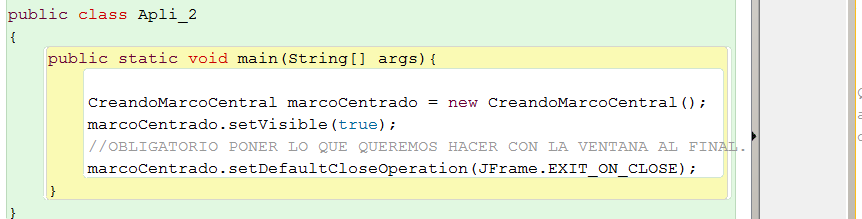
***COLOCA EL FRAME EN EL CENTRO DE LA PANTALLA***

***QUITAMOS EL ICONO DE JAVA QUE SIEMPRE APARECE CON LAS VENTANAS.***

Para colocar el frame en el centro de la pantalla se necesita concer el tamaño de la pantalla.

Para ello java dispone de la clase Toolkit, donde se almacenan muchos métodos que lo que hacen es comunicarse con el sistema nativo de ventanas, es capaz de determinar el tamaño de la ventana donde se esté ejecutando.





VIDEO VIDEO 58

***AGREGAR CONTENIDO AL FRAME O MARCO.***

Se puede agregar imágenes, texto, figuras geométricas, buzones etc

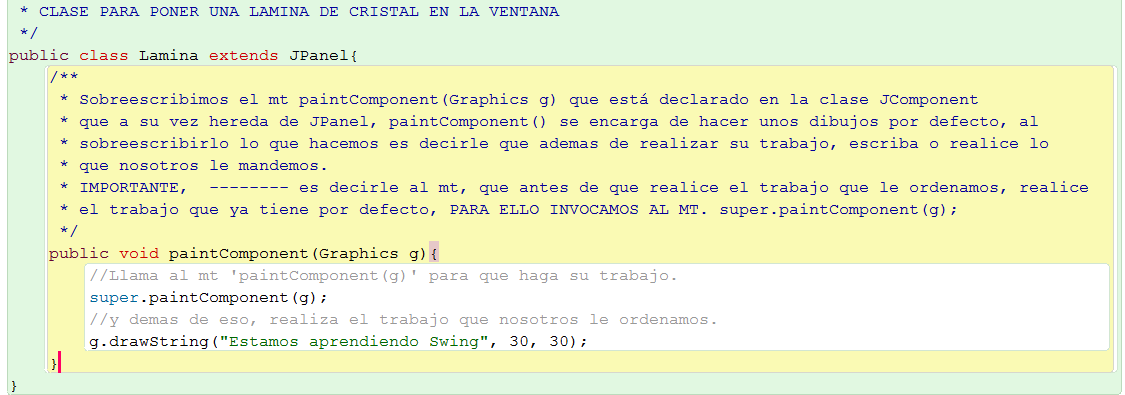
Se puede escribir directamente sobre el frame, pero no es recomendable, tenemos que verlo como la mesa de un dibujante, que pone un folio encima de la mesa para dibujar sobre él, no sobre la mesa.

A los frames no se les añade folios, se les añade paneles o láminas (cristal trasnparente donde dibujar o poner botones etc), se pueden poner varios cristales transparentes.

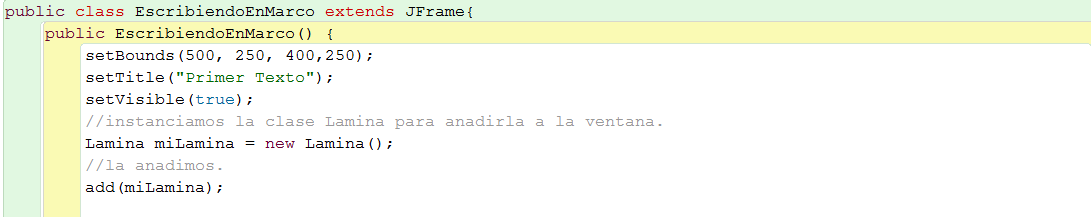
PARA ESTO SE UTILIZA LA CLASE JPanel, esta clase tiene cientos de mt.

Se crea una clase que herede de JPanel y ya disponemos de todos sus mt.

* En esta clase invocamos al mt paintComponente(Graphics g) de la clase [javax.swing.JComponent](file:///C:\Users\Usuario\Desktop\CURSO_JAVA\00%20Todas_Las_Clases_De_Java\api\javax\swing\JComponent.html) de la que hereda JPanel.

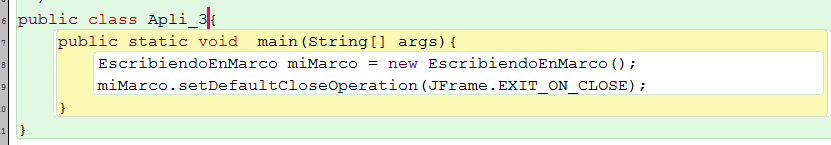


Una vez hecho esto, cuando instancias esta clase desde la clase donde creamos la ventana, disponemos de todos los métodos que dispone Jpanel y la añadimos a la ventana.



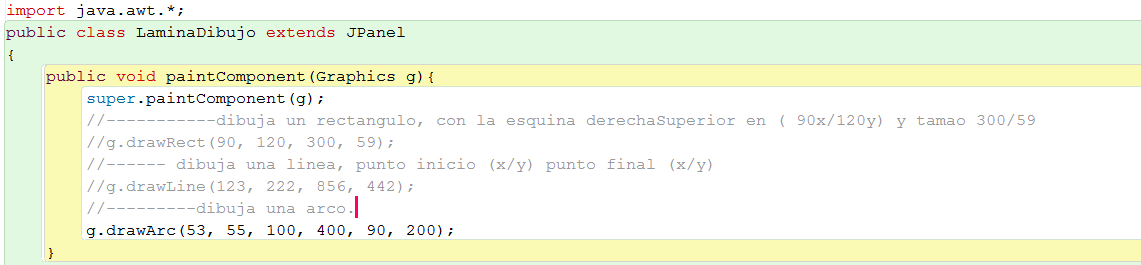
. ***minuto 6;***  **paintComponent(Graphics g)** es el método que dibuja los componentes (buzones, figuras geométricas, imagenes etc) encima de los paneles, **Graphics g** este paramétro pertenece a la clase Graphis, la cual nos ofrece todas las herramientas para dibujar gráficos (texto, figuras geométricas etc.

Ejecutamos la aplicacion.



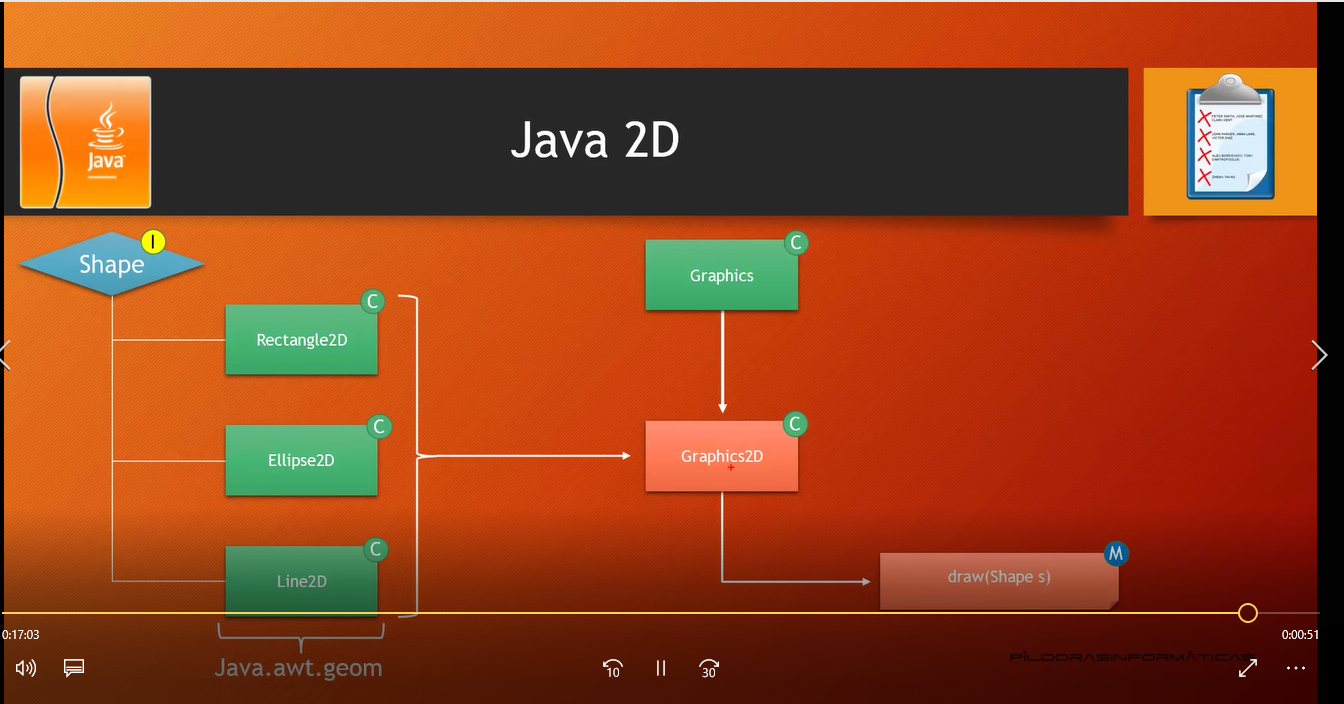
VIDEO VIDEO 59

***PARA HACER DIBUJOS SENCILLOS.***



VIDEO VIDEO 60

***PARA HACER DIBUJOS MAS EFICIENTES.***

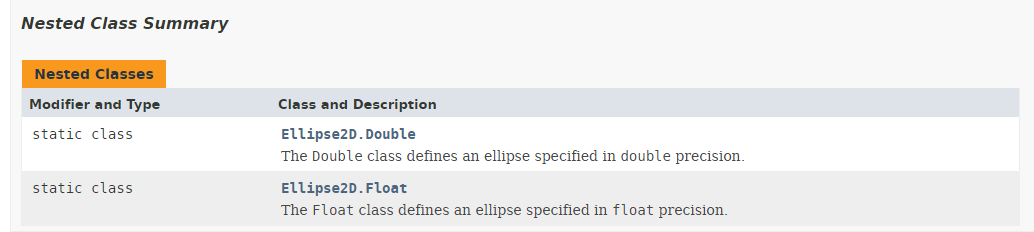
***.***

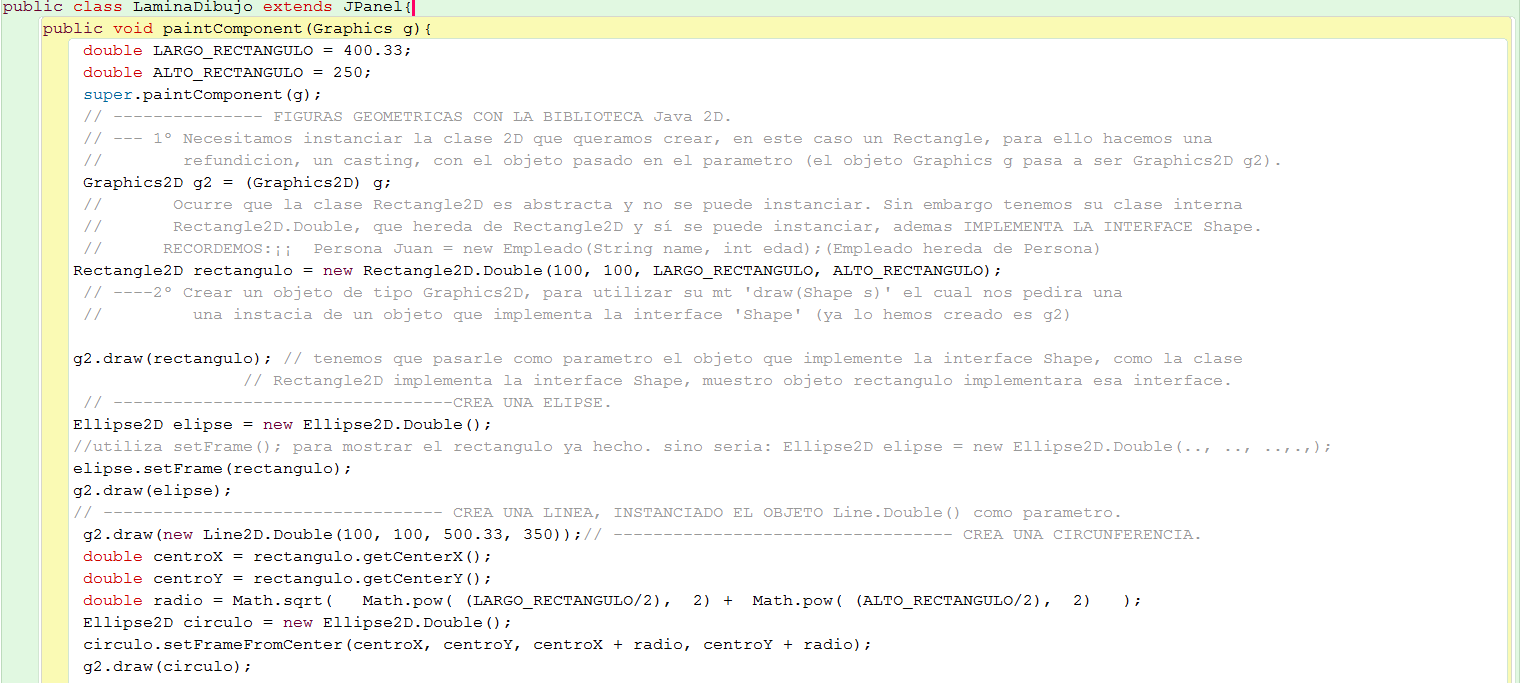
Para realizar este tipo de dibujos tenemos que acceder a la **biblioteca ‘Java 2D** , dispone de varias clases , Rectangle2D, Elipse2D etc, estas clases implementa la **interface ‘Shape’** y pertenecen al **paquete ‘java.awt.geom’.** Para poder utilizar estas clases necesitamos un **objeto de tipo Graphics2D** en vez de un objeto de tipo **Graphics** como en el video anterior.

La **clase Graphics2D** tiene el mt **draw(Shape s)** que pide como parámetro un objeto **Shape**, es

***NOTA:***

Todas las **clases 2D son abstractas**, no se pueden instanciar, pero todas ellas tiene clases internas que heredan de ellas y sí se pueden instanciar..





VIDEO VIDEO 61

***PONER COLOR EN LAS FIGURAS***

Tenemos que tener en cuenta el **mt ‘setPaint(Color )**’ de la clase **Graphics2D**, y la clase ‘**Color**’ con sus diferentes constantes de COLOR.



, **setBackground(Color)** establece un color d fondo para la lamina.

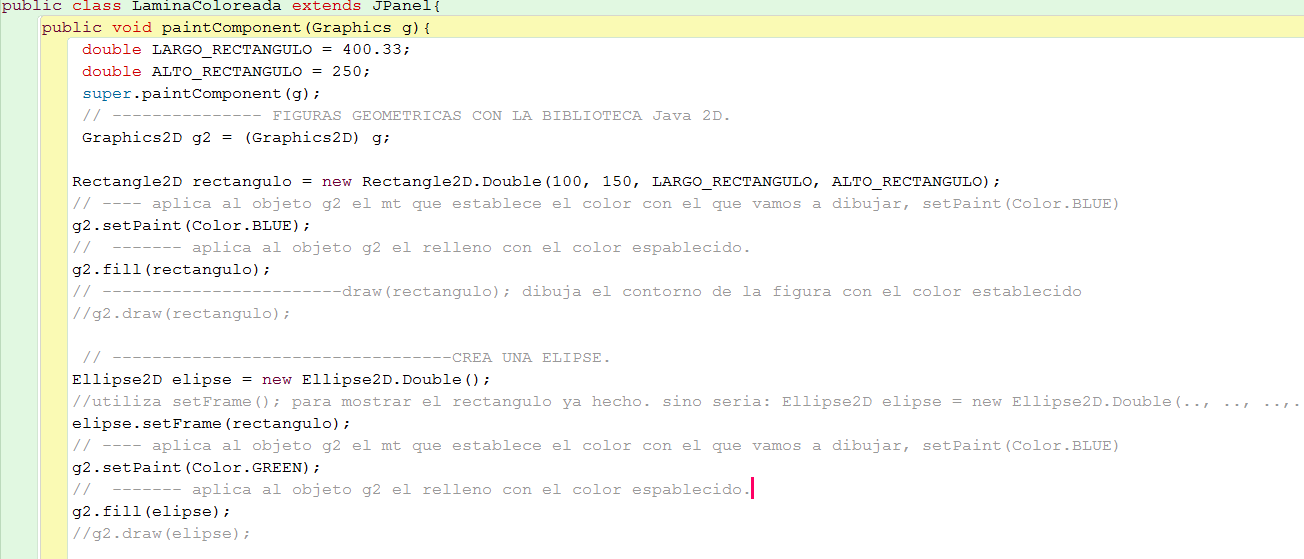
, **setForeground(Color)** establece un color frontal, con el que despues vas a dibujar todo lo que aparezca en la lamina pertenecen a la clase Component y los hereda la clase Jpane.-

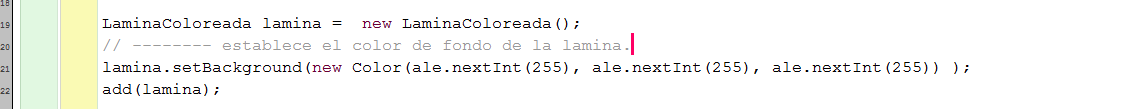
// ---- aplica al objeto g2 el mt que establece el color con el que vamos a dibujar, setPaint(Color.BLUE)

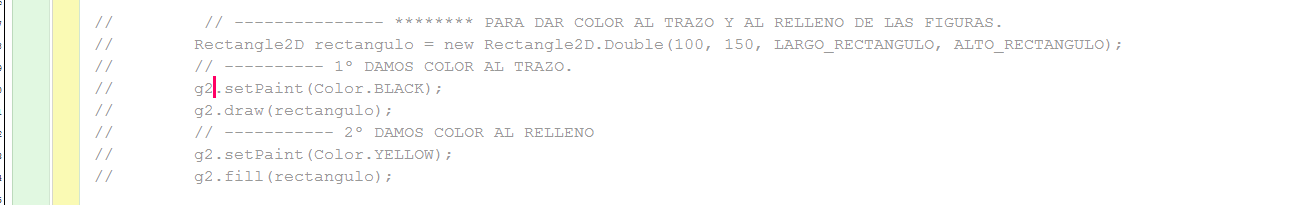
g2.setPaint(Color.BLUE);

DATO:

El mt **draw();** dibuja el contorno de la figura, el mt **fill();** dibuja el relleno.





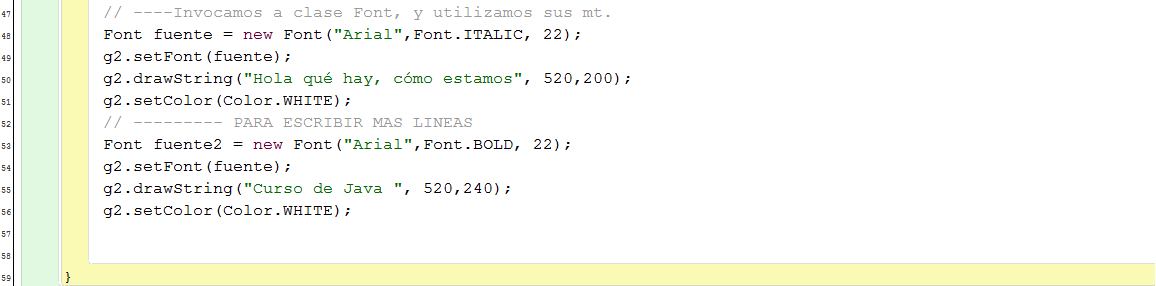


VIDEO VIDEO 62

***TIPO DE LETRA, FUENTE***

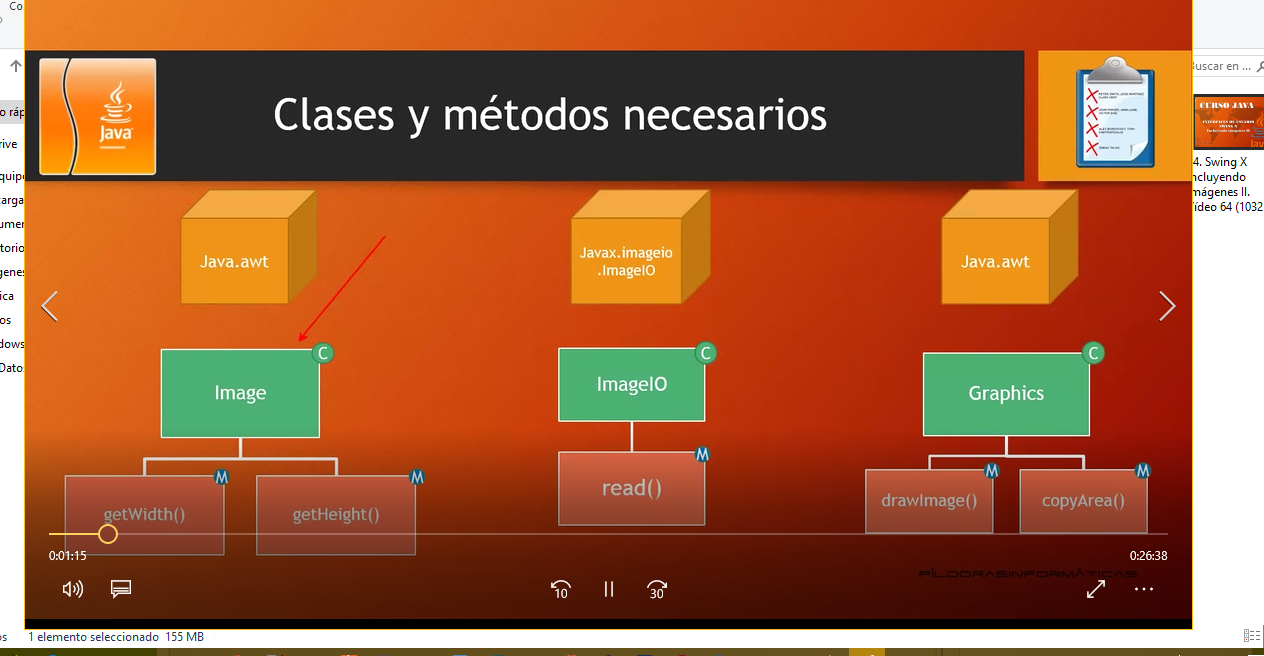
Importamos la clase Font , esta en el paquete ‘awt’ y aplicamos sus métodos, en la clase Lamina

setFont().



VIDEO VIDEO 63

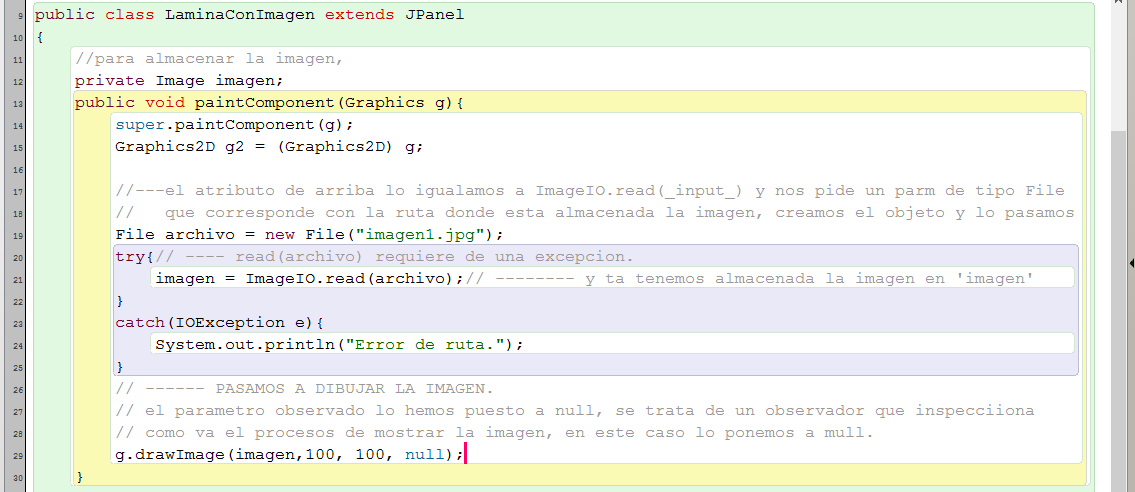
***MANIPULACIÓN DE IMÁGENES DENTRO DE UN MARCO.***



La **clase Image** para poder almacenar la imagen en una variable de tipo **Image**, **getWidth()** devuelve el ancho de la imagen, **getHeight()** el alto.

La **clase ImageIO** para rescatar imagense que se encuentren fuera del programa, en alguna carpeta o en alguna URL, para ello utilizaremos el **mt read().**

El mt copyArea() determina un area de la lámina para colocar la imagen.



VIDEO VIDEO 64

***PARA COPIAR UNA MISMA IMAGEN A LO LARGO Y ANCHO DE LA VENTANA.***

Utilizamos el **mt ‘copyArea()’** aplicado a **‘g’** que es el parámetro pasado al mt.

------------------------------------------------------------  **g.copyArea(0, 0, largo, alto, 250, 0);**

//repite la imagen situada en las coordenadas (0/0), con un tamano de (200/200 de la imagen

//que se copia), en las coordenadas (250/0)

PARA **COPIAR LA IMAGEN VARIAS VECES** HEMOS DE UTILIZAR BUCLES PARA QUE LA IMAGEN VAYA REPITIENDOSE.

ANTES DE ESTO AVERIGUAMOS EL TAMAÑO DE LA IMAGEN.

