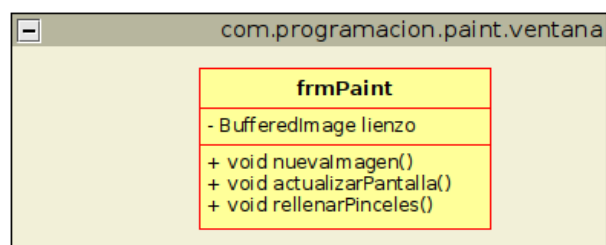
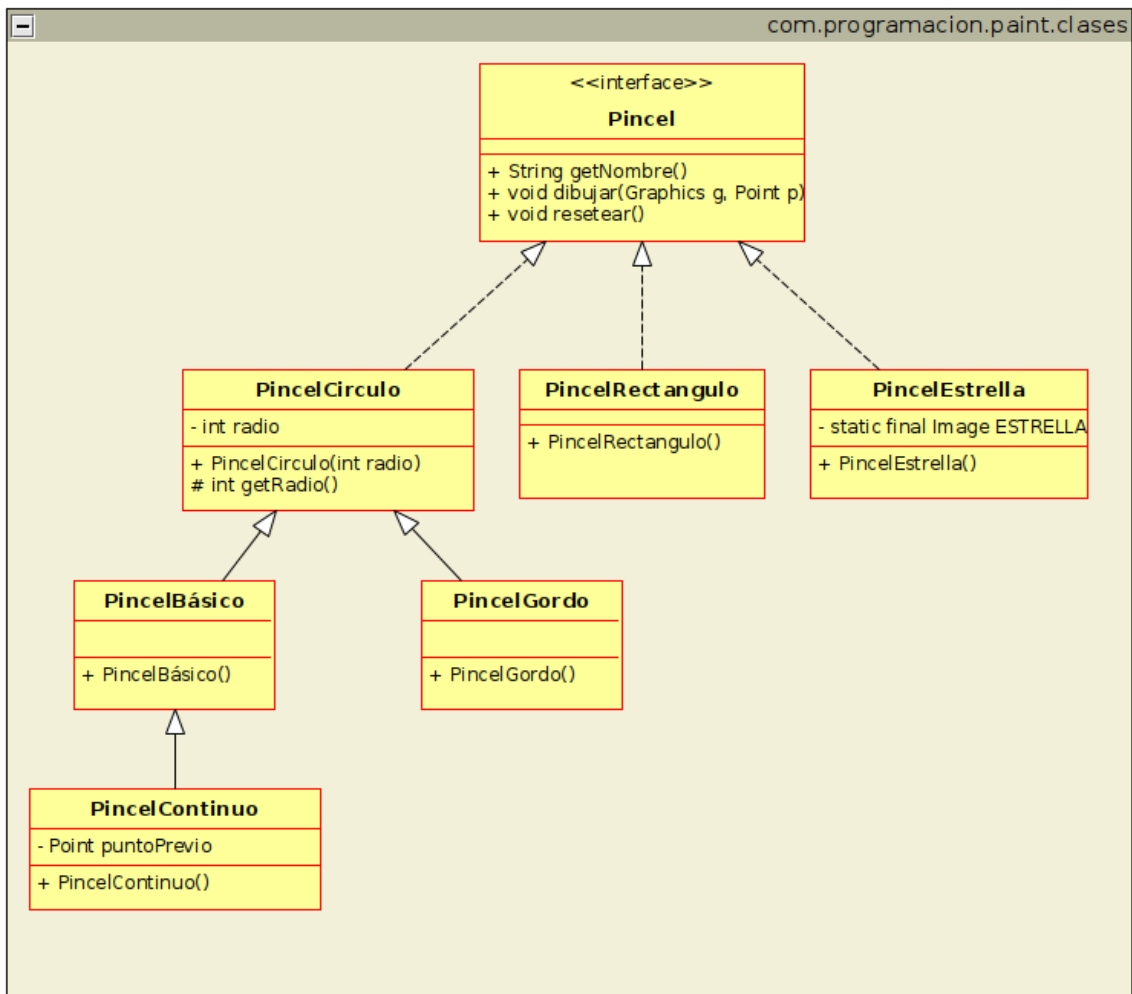
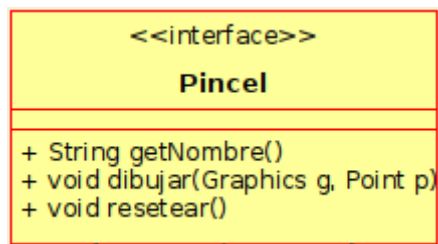


PROYECTO 5: PAINT





Esta interfaz representa un pincel con el que podemos dibujar.

Métodos (a implementar en las clases):

- **getNombre:** Devuelve el nombre del pincel.
- **dibujar:** Este método utiliza el Graphics pasado como primer parámetro para dibujar un punto (la forma de dibujar será programada en las clases hijas) en las coordenadas pasadas como segundo parámetro.
- **resetear:** Este método reinicia el estado del pincel a su forma inicial, según sea indicado en la documentación de las clases hijas. Este método es útil cuando se cambia de un pincel a otro en el programa.

PincelCirculo
- int radio
+ PincelCirculo(int radio) # int getRadio()

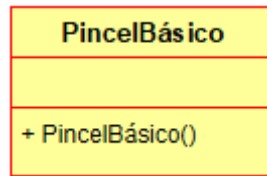
Esta clase es un pincel cuya pluma tiene forma de círculo y por tanto, cuando lo elijamos veremos que nuestro trazo son círculos huecos (no hay que rellenarlos de color).

Propiedades:

- **radio:** Es el radio del pincel

Métodos:

- **Constructor:** Crea un pincel cuyo radio se pasa como parámetro
- **getNombre:** Devuelve el texto "Pincel con pluma circular de radio *(hacer que aquí aparezca el valor de la propiedad radio)*"
- **getRadio:** Devuelve el radio del pincel.
- **dibujar:** Usa el Graphics pasado como primer parámetro para dibujar un círculo del radio que tiene el pincel en el punto que se pasa como segundo parámetro.
- **resetear:** No hace nada
- **toString:** Devuelve el nombre del pincel



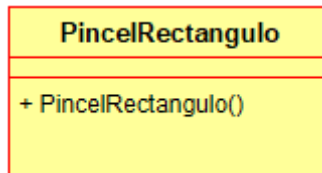
Esta clase es un pincel cuya pluma es un círculo de radio 1, por lo que veremos que nuestro trazo son puntos.

- **Constructor:** Crea un pincel cuyo radio es 1
- **getNombre:** Devuelve el texto "Pincel básico"
- **toString:** Devuelve el nombre del pincel

PincelGordo
+ PincelGordo()

Esta clase es un pincel cuya pluma es un círculo relleno de color de radio 20, por lo que veremos que nuestro trazo es un círculo coloreado (el color viene ya puesto en el Graphics)


- **Constructor:** Crea un pincel cuyo radio es 20
- **getNombre:** Devuelve el texto “Pincel gordo”
- **dibujar:** Es necesario sobrescribir este método de esta forma:
 - Usa el método getColor del Graphics pasado como parámetro para saber el color con el que hay que dibujar el círculo.
 - Usa el Graphics para dibujar un círculo relleno de dicho color en las coordenadas que indica el objeto Point pasado como segundo parámetro.
- **toString:** Devuelve el nombre del pincel



Esta clase es un pincel cuya pluma tiene forma de rectángulo y por tanto, cuando lo elijamos veremos que nuestro trazo son rectángulos huecos (no hay que rellenarlos de color).

- **getNombre:** Devuelve el texto “Pincel de pluma rectangular”
- **getRadio:** Devuelve el radio del pincel.
- **dibujar:** Usa el Graphics pasado como primer parámetro para dibujar un rectángulo sin rellenar de color, en la posición indicada por el segundo parámetro. El rectángulo tendrá 15 píxeles de alto y de ancho.
- **resetear:** No hace nada
- **toString:** Devuelve el nombre del pincel.

PincelEstrella
- static final Image ESTRELLA
+ PincelEstrella()

Esta clase es un pincel cuya pluma tiene esta forma:  y cuando lo elijamos, veremos que nuestro trazo son estrellas como esa.

Propiedades:

- **ESTRELLA:** Es una constante que guarda el dibujo de la estrella. Esta propiedad se rellena en el bloque inicializador estático de la clase.

Métodos:

- **Bloque inicializador estático:** Deberá cargar la imagen del archivo e inicializar la constante ESTRELLA.
 - *Si no recuerdas lo que era un bloque inicializador estático, consulta el apartado “propiedades estáticas” de los apuntes del tema 4*
- **getNombre:** Devuelve el texto “Pincel estrella”
- **dibujar:** Usa el Graphics para dibujar la estrella en las coordenadas que se pasan como segundo parámetro.
- **resetear:** No hace nada
- **toString:** Devuelve el nombre del pincel.

PincelContinuo
- Point puntoPrevio
+ PincelContinuo()

Es un pincel que dibuja un trazo continuo (a diferencia de los otros pinceles, en los que se ven puntos cuando se dibuja). Para ello, este pincel “recuerda” cuáles fueron las coordenadas del último punto dibujado, para así unirlos con una línea recta al nuevo punto. De esta forma se obtiene un trazo continuo.

Propiedades:

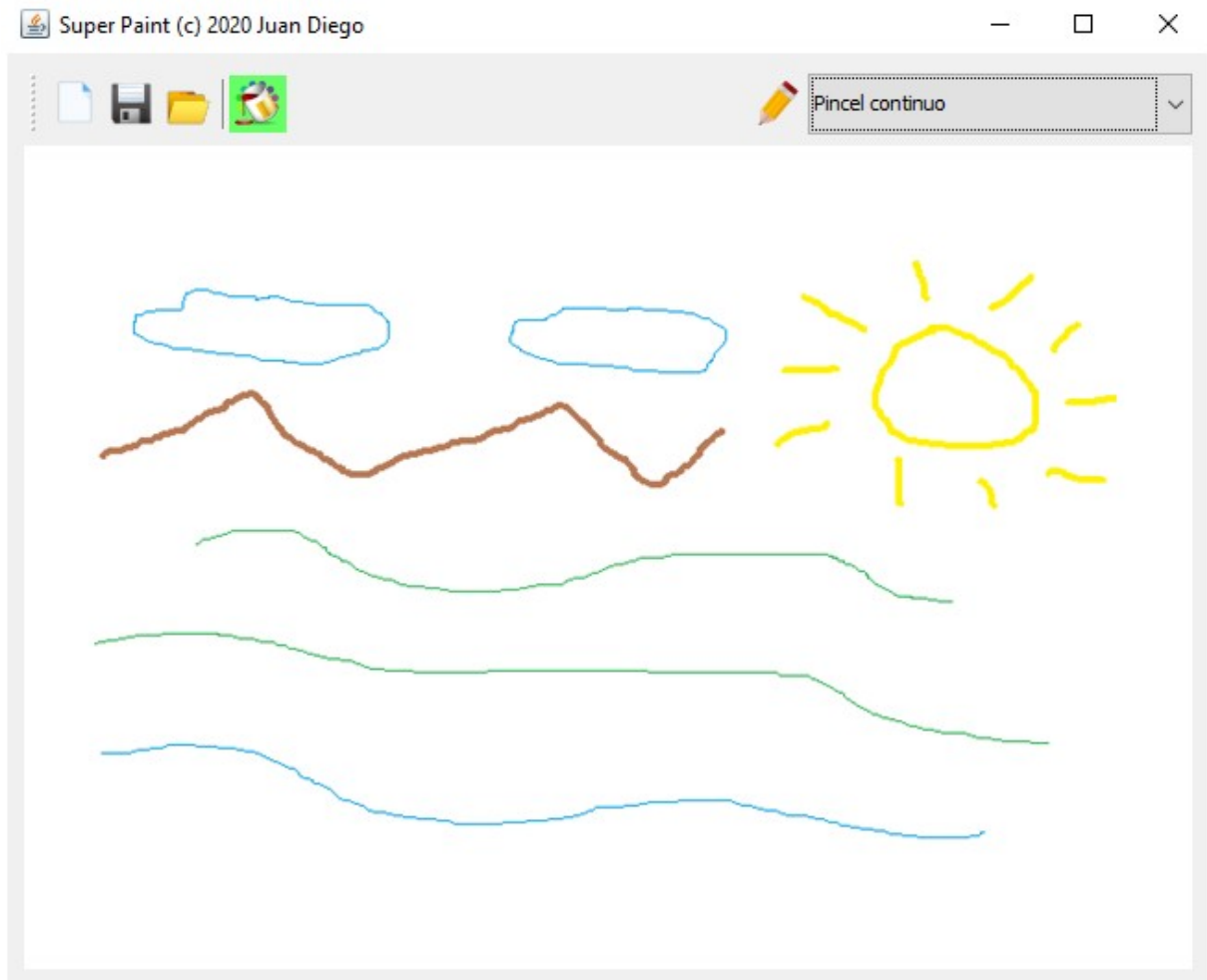
- **puntoPrevio:** Es un objeto que guarda las coordenadas del último punto dibujado.

Métodos:

- **Constructor:** Crea un pincel cuyo **puntoPrevio** es null (esto es porque inicialmente no se ha dibujado nada).
- **getNombre:** Devuelve el texto “Pincel continuo”
- **dibujar:** Funciona así:
 - Si **puntoPrevio** es null, dibuja un punto en las coordenadas pasadas como segundo parámetro
 - Si **puntoPrevio** no es null, entonces usa el Graphics pasado como parámetro para dibujar una línea recta desde el punto previo hasta el Point pasado como segundo parámetro.
 - En ambos casos, el método terminará actualizando la propiedad **puntoPrevio** para que guarde el punto que se ha pasado como segundo parámetro.
- **resetear:** Guarda null dentro de la propiedad **puntoPrevio**

frmPaint
- BufferedImage lienzo
+ void nuevaimagen() + void actualizarPantalla() + void rellenarPinceles()

Esta es la ventana de un programa que sirve para dibujar con el ratón:



1) Cómo dibujar la ventana

- Crea un **JFrame** llamado frmPaint dentro del paquete com.programacion.paint.ventanas
- Crea un paquete llamado com.programacion.paint.recursos y copia dentro de él las imágenes de los iconos que te ha dado el profesor.
- Desmarca la propiedad **resizable** de la ventana
- Arriba a la izquierda arrastra un **JToolBar**, que es un contenedor de botones y llámalo **jtbBotones**

- Arrastra dentro de **jtbBotones** cuatro **JButton**
 - Llama a los botones **cmdNuevo**, **cmdGuardar**, **cmdAbrir** y **cmdColor**
 - En la propiedad **icon** de cada botón pon la imagen correspondiente (las tienes en el paquete de recursos que has agregado) y borra el texto de los botones para que se queden solo los iconos.
 - Asegúrate de que **cmdColor** tenga la propiedad **opaque** marcada
 - Pon en la propiedad **background de cmdColor** el color negro. Deberá verse el fondo del botón de negro.
- Arriba a la derecha arrastra un **JLabel** y ponle el icono del lápiz
- Junto al lápiz, añade un **JComboBox** y llámalo **cmbPinceles**
 - En las propiedades de **cmbPinceles**, busca **model** y borra todo lo que NetBeans te pone por defecto.
 - En la sección **Code** de **cmbPinceles**, busca **Type Parameters** y escribe **<Pincel>**
- Arrastra en la zona central de la ventana un **JLabel**
 - Llámalo **lblAreaDibujo**
 - Marca su propiedad **opaque**
 - Ponle de color de fondo (propiedad **background**) negro
 - Cambia su propiedad **cursor** a **Cursor de punto de mira**

2) Código fuente de la ventana

- Añade la propiedad **lienzo** a la ventana y ponla a **null** en el constructor.
 - Añade a la ventana un método **private void nuevalmagen()** y prográmalo así:
 - Inicializa la propiedad **lienzo** con una **BufferedImage** que tenga el mismo ancho y el mismo alto que la imagen **lblAreaDibujo** y cuyo tipo sea la constante **BufferedImage.TYPE_INT_RGB**
 - Llama al método **actualizarPantalla** que vas a programar a continuación.
 - Añade otro método privado llamado **private void actualizarPantalla()** de esta forma:
 - Usa el método **getGraphics** del objeto **lblAreaDibujo** para obtener un “lápiz” para dibujar sobre él.
 - Consulta la documentación del **Graphics** para conseguir dibujar la imagen que hay en la propiedad **lienzo** en las coordenadas (0,0). Con esto dibujamos **lienzo** dentro de **lblAreaDibujo**.
- **Nota:** Puedes pasar **null** cuando veas que la documentación te pida un **ImageObserver**

- Añade a la ventana un método privado **private void rellenarPinceles()**
 - Dentro de ese método rellena el combo box con los siguientes objetos:
 - Un pincel básico
 - Un pincel continuo
 - Un pincel gordo
 - Un pincel rectángulo
 - Un pincel estrella
- Selecciona la ventana y programa su manejador del evento **windowOpened** (ese evento se lanza cada vez que se abre la ventana por primera vez), dentro de él llama al método **rellenarPinceles** y justo después, al método **nuevalmagen**
- Selecciona **IblAreaDibujo** y programa su evento **MouseDragged** (ese evento se lanza cuando pulsas el ratón y lo arrastras sobre el área de dibujo) de esta forma:
 - Usa el método **getSelectedItem()** del combobox para sacar un Object con el pincel seleccionado.
 - Haz un **casting** a ese Object para convertirlo en un objeto **Pincel**. De esta forma se obtiene el objeto Pincel seleccionado en el combobox.
 - Obtén el Graphics de la propiedad **lienzo** con su método **getGraphics**
 - Obtén el color de fondo que tiene el botón **cmdColor**
 - Pon a ese graphics el color que acabas de recuperar.
 - Obtén un objeto que guarda las coordenadas del ratón usando el método **getPoint** del objeto **evt** que recibe como parámetro el método **IblAreaDibujoMouseDragged** (debe ser el método en el que te encuentras programando).
 - Llama al método **dibujar** del pincel
 - Llama al método **actualizarPantalla**
- Selecciona el combobox **cmbPinceles** y programa su manejador del evento **itemStateChange** (este evento se lanza cada vez que cambia el elemento seleccionado del combobox) así:
 - Obtén el pincel seleccionado, tal y como has hecho anteriormente
 - Llama al método resetear del pincel que has obtenido. Con esto, reiniciamos el pincel cada vez que el usuario cambia de pincel.

- Haz que al pulsar **cmdColor** :
 - Crea un **JColorChooser** usando su constructor sin parámetros.
 - Un JColorChooser es una ventana ya programada en Java que permite al usuario elegir un color.
 - Llama al método **setVisible** del **JColorChooser** para que se vea.
 - Usa el método **getColor** del **JColorChooser** para recuperar el color seleccionado.
 - Usa el método **setBackground** de **cmdColor** para cambiar el color de fondo del botón y ponle el color seleccionado
- Haz que al pulsar **cmdNuevo** se llame al método **nuevalimagen**
- Haz que al pulsar **cmdGuardar**
 - Se abra un **JFileChooser** que nos permita elegir un archivo para guardar el dibujo
 - Use la clase **ImageIO** para guardar **lienzo** en formato JPG.
 - Muestra una ventana emergente informando del resultado de la operación (guardado correctamente o error al guardar).
- Haz que al pulsar **cmdAbrir**
 - Se abra un **JFileChooser** que nos permita elegir un archivo
 - Usa la clase **ImageIO** para cargar la imagen en la propiedad **lienzo** y después llamar al método **actualizarPantalla**
 - En caso de no poder abrirse la imagen se mostrará una ventana emergente informando del resultado.