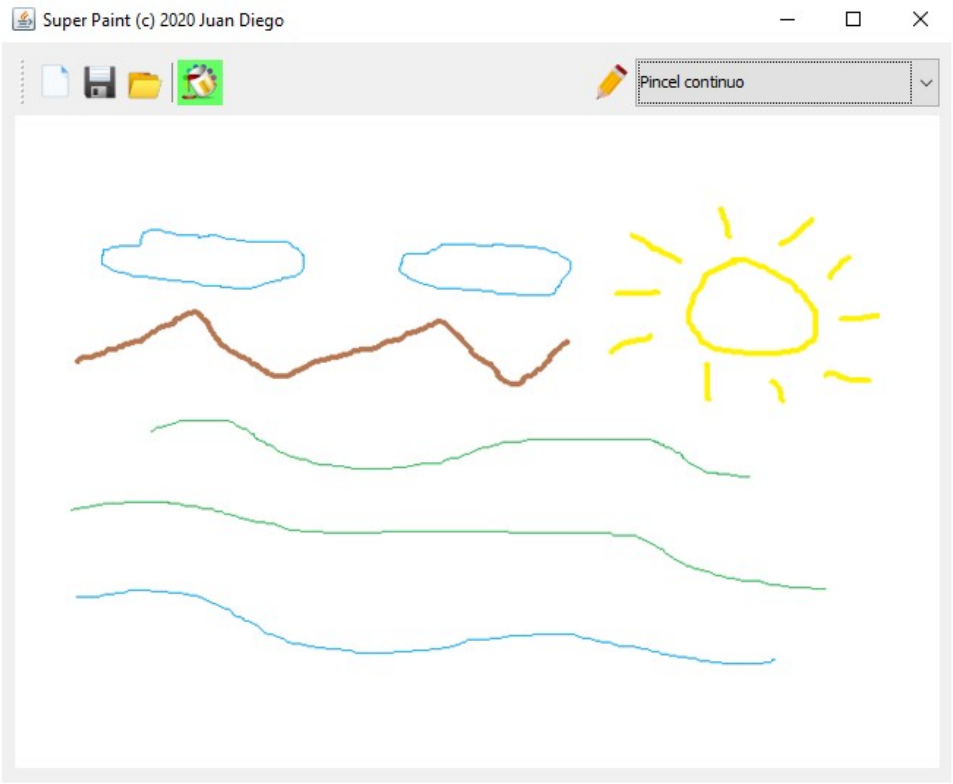
PROGRAMA 4: PAINT (VERSIÓN JAVA 8)

Este programa consiste en hacer un programa sencillo para dibujar con el ratón:



1) Cómo dibujar la ventana

- Crea un proyecto y añade un paquete llamado **entornos.programa4.ventanas**
- Crea un **JFrame** para la ventana del programa en el paquete.
- Desmarca la propiedad **resizable** de la ventana
- Arriba a la izquierda arrastra un **JToolBar**, que es un contenedor de botones.
    - Llámalo **jtbBotones**
- Arrastra dentro de **jtbBotones** cuatro **JButton**
    - Llama a los botones **cmdNuevo, cmdGuardar, cmdAbrir** y **cmdColor**
    - Pon a los botones las imágenes que se ven y borra su texto
    - Asegúrate de que **cmdColor** tenga la propiedad **opaque** marcada
    - Ponle a **cmdColor** el color de fondo negro (propiedad **background**)
- Arriba a la derecha arrastra un **JLabel** y ponle de icono una imagen con un 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 color de fondo (propiedad **background)** negro
    - Cambia su propiedad **cursor** a **Cursor de punto de mira**

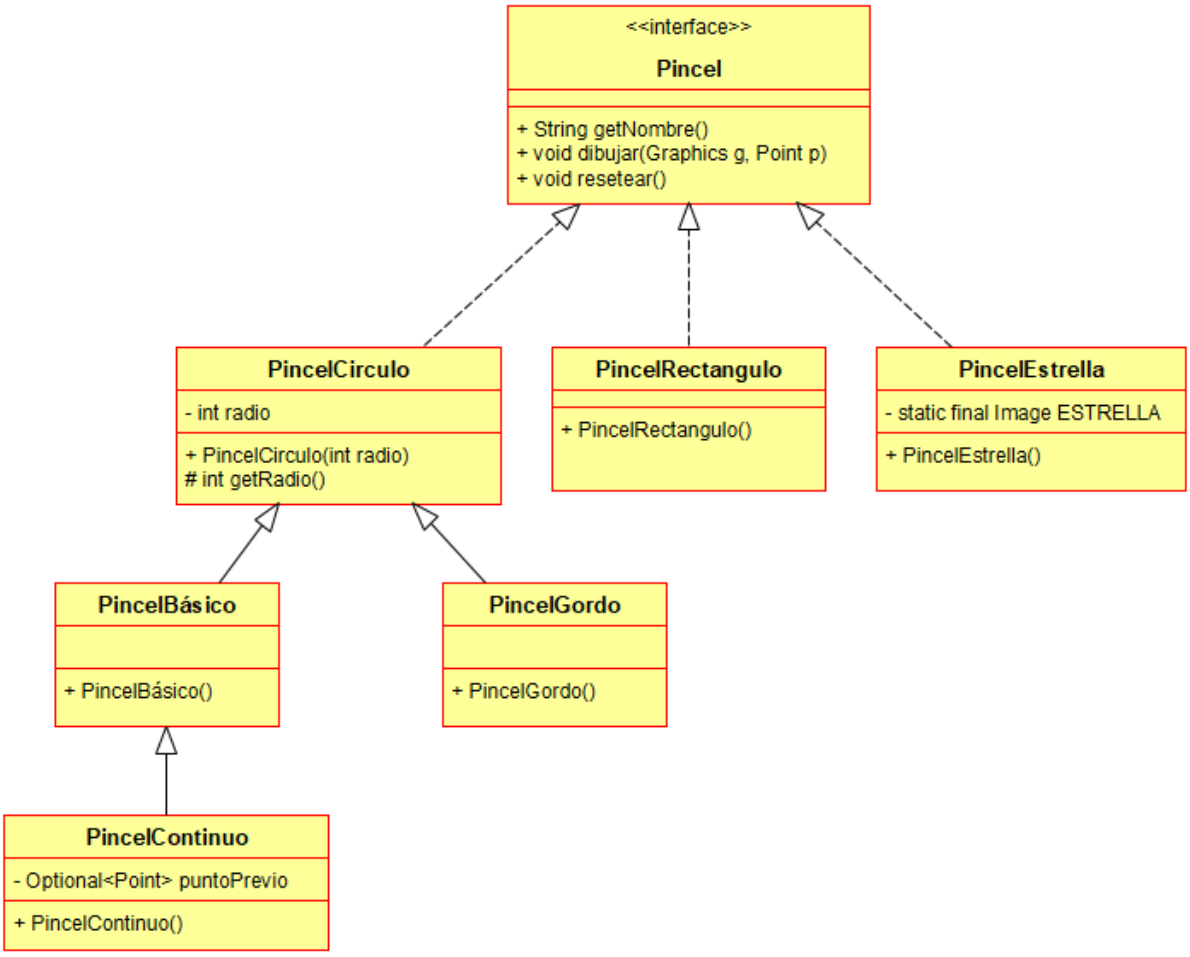
2) Cómo hacer el programa

Este programa tiene dos partes que tendrás que hacer en este orden:

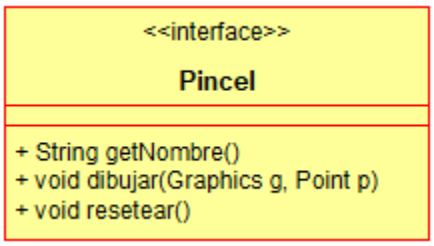
- En primer lugar tienes que programar el diagrama de clases del siguiente apartado
- A continuación, podrás programar la ventana, usando las clases del diagrama

3) El diagrama de clases

Deberá hacerse en un paquete llamado **entornos.programa4.pinceles**



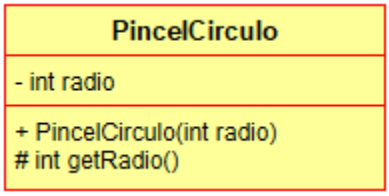
3.1) La interfaz Pincel



Esta interfaz representa un pincel que puede ser usado en nuestro programa para dibujar. Sus métodos son:

- **getNombre:** Devuelve el nombre del pincel
- **dibujar:** Hace que el pincel haga una marca en el punto que se pasa como parámetro ayudándose para ello en el objeto Graphics pasado como parámetro.
- **resetear:** Reinicia el estado interno del pincel (esto es útil cuando se cambia de un pincel a otro en el programa)

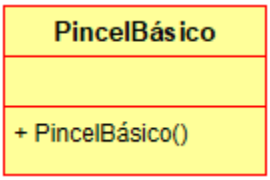
3.2) La clase PincelCirculo



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).

- **Constructor:** Crea un pincel cuyo radio se pasa como parámetro
- **getNombre**: Devuelve el texto “Pincel con pluma circular de 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

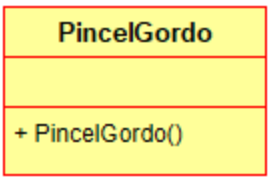
3.3) La clase PincelBásico



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”

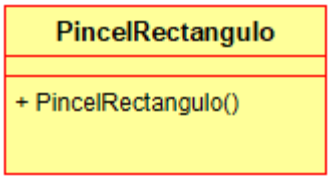
3.4) La clase PincelGordo



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 reprogramar este método heredado para que el círculo se dibuje rellenado del color que tenga el Graphics en ese momento.

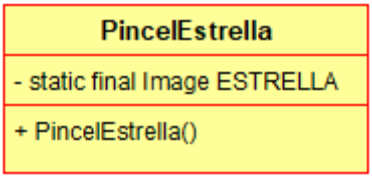
3.5) La clase PincelRectangulo



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

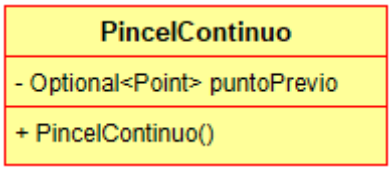
3.6) La clase PincelEstrella



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

- **Bloque inicializador estático:** Deberá cargar la imagen del archivo e inicializar la constante ESTRELLA.
- **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

3.7) La clase PincelContinuo



Es un pincel que “recuerda” cuáles fueron las coordenadas del último punto dibujado, para unirlas con una línea recta al nuevo punto. De esta forma se obtiene un trazo continuo.

- **Constructor:** Crea un pincel cuyo **puntoPrevio** es un Optional vacío.
- **getNombre:** Devuelve el texto “Pincel continuo”
- **dibujar:** Funciona así:
  - Si **puntoPrevio** no tiene nada dentro, dibuja un punto en las coordenadas pasadas como segundo parámetro
  - Si **puntoPrevio** guarda un punto, entonces dibuja una línea recta desde el punto previo hasta el punto pasado como segundo parámetro
  - En ambos casos, la propiedad **puntoPrevio** se actualizará para guardar el punto que se pasa como segundo parámetro.
- **resetear:** Vuelve a poner **puntoPrevio** con un Optional vacío.

4) El programa

Sigue estas instrucciones para realizar el programa:

- Abre el código fuente de la ventana, que como puedes ver al principio de su código fuente es una clase
    - En este ejercicio vamos a aprovechar que la ventana es una clase y le vamos a añadir nuestras propiedades y nuestros métodos.
- Añade una propiedad **private BufferedImage** a la ventana y llámala **lienzo**
- Añade a la ventana un método **private void nuevaImagen()** 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.
    - *Ampliación: Si quieres, al principio del método puedes usar un **JColorChooser** para que aparezca una ventana que te deje elegir un color. Puedes usar ese color para rellenar la imagen **lienzo**. Si no, el fondo de la imagen será negro. La clase JColorChooser está explicada más abajo.*
- Añade otro método privado llamado **private void actualizarPantalla()** de esta forma:
    - Obtén el Graphics de **lblAreaDibujo**
    - Dibuja la imagen que hay en la propiedad **lienzo** en las coordenadas (0,0) del Graphics que acabas de obtener. Con esto dibujamos **lienzo** dentro de **lblAreaDibujo**.
        - Puedes pasar null cuando 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 **nuevaImagen**
- Selecciona **lblAreaDibujo** 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 verlo como un objeto Pincel, obteniendo así un objeto de Pincel que guarde el pincel seleccionado.
    - 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 las coordenadas del ratón con el método **getPoint** del objeto **evt** que viene con el manejador del evento.
    - Llama al método **dibujar** del pincel
    - Llama al método **actualizarPantalla**
- Selecciona el combobox **cmbPinceles** y programa su manejador del evento **itemStateChange** (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
    - Llama al método **setVisible** del **JColorChooser** para que se vea
    - Usa el método **getColor** del **JColorChooser** para recuperar el color seleccionado
    - Pon de color de fondo de **cmdColor** el color seleccionado, con su método **setBackground**
- Haz que al pulsar **cmdNuevo** se llame al método **nuevaImagen**
- 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.
    - Se mostrará una ventana emergente informando del resultado de la operación
- 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.